UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
Departamento de Informática

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

Estudio y desarrollo de un escritorio en "la nube" basado en FreeNX y Dropbox

Autor: Antonio López Vivar
Tutor: Alejandro Calderón Mateos

Leganés, julio de 2011

"Sed fugit interea, fugit irreparabile tempus,
singula dum capti circumvectamur amore."

Virgilio, Geórgicas, III, 284-285

Agradecimientos

A mis padres Antonio y Milagros, a quienes debo muchísimo más de lo que jamás podré devolverles.

A mi hermana y mejor amiga Aurora, un espejo donde mirarse cada día.

A mis ex compañeros de trinchera, especialmente a Javier y Mayca, cuya amistad es el activo más valioso que he conseguido en estos inolvidables años.

A mi tutor Alejandro, por sus certeros golpes de timón que contribuyeron a que este "barco" llegara a buen puerto.

A todos los que alguna vez se esforzaron en enseñarme algo y en especial a aquellos que lo consiguieron.

Resumen

En este documento se aborda el desarrollo de una aplicación multiplataforma con interfaz web que permita virtualizar de forma remota aplicaciones permitiendo además la persistencia de los ficheros creados en "la nube".

Primeramente se muestra un estudio de los distintos protocolos de escritorio remoto existentes, profundizando más en NX, la opción elegida.

A continuación se lleva a cabo el análisis de un servicio real muy similar en propósito a Clouded Desktop así como una comparativa entre ambos.

Por último se refleja el análisis, diseño y los detalles más significativos de implementación de Clouded Desktop.

Abstract

This paper tackles the development of a multiplatform application with a web interface that allows remote virtualization of applications and persistence of the files created in "the cloud".

First it will be shown a study of different existing remote desktop protocols, going deeper in NX, the option chosen.

The following is an analysis of a real service that operates nowdays and a comparison between it and Clouded Desktop.

Finally, it is shown the analysis, design and most significant implementation details of Clouded Desktop.

Índice general

Índice de figuras

Índice de tablas

1 Introducción y objetivos

1.1 Introducción

La motivación de este proyecto parte del deseo de utilizar aplicaciones de distintas plataformas sin tener que preocuparnos de:

- Requisitos de software (sistema operativo determinado, librerías, etc.).
- Requisitos de hardware (CPU, memoria principal, espacio en disco, etc.).
- Instalación de la aplicación.
- Mantenimiento de la aplicación.

Hasta hace unos años si se deseaba ejecutar aplicaciones de una plataforma distinta a la nuestra era necesario particionar el disco duro e instalar otro sistema operativo y después las aplicaciones deseadas.

Más tarde, apareció software de virtualización que facilitaba la tarea anterior, ya que permitía la creación de máquinas virtuales dentro de nuestro sistema operativo donde instalar otros sistemas operativos con sus aplicaciones.

La idea de este proyecto es ir un paso más allá, evitando al usuario tener que instalar o mantener actualizado más software de ningún tipo. Podrá centrarse simplemente en utilizar las aplicaciones deseadas (independientemente de su plataforma) y para ello lo único que necesitará será su navegador web favorito y una conexión a Internet.

1.2 Objetivo

El objetivo por tanto es desarrollar una aplicación multiplataforma que permita la virtualización remota de aplicaciones dando la sensación al usuario de que se ejecutan en su equipo ("seamless") cuando en realidad lo están haciendo en el servidor de Clouded Desktop.

Además, todos los ficheros creados/borrados/modificados con las aplicaciones de Clouded Desktop serán sincronizados en "la nube" mediante el uso transparente de Dropbox.

Para todo esto se hará uso de una interfaz web que simulará el escritorio típico de un sistema operativo donde el usuario dispondrá de una lista de aplicaciones para utilizar pudiendo confeccionarse su propia lista personalizada de aplicaciones favoritas.

1.3 Fases del desarrollo

En la primera fase del desarrollo se llevó a cabo un estudio sobre los distintos protocolos de escritorio remoto investigando sus características en busca del más eficiente y que al mismo tiempo sirviera para el objetivo buscado.

En la siguiente fase, una vez elegido el protocolo NX se diseñó e implementó una versión muy primitiva del "back-end" de Clouded Desktop para estudiar la viabilidad real.

Comprobado que la funcionalidad era correcta, se completó el diseño y la implementación del back-end y comenzó el análisis, diseño e implementación de la interfaz web. Durante esta fase se realizaron a su vez pruebas en busca de errores y para optimizar el funcionamiento del sistema.

La confección de esta memoria se fue realizando progresivamente, aunque se solapó en el

tiempo con las últimas pruebas y mejoras del prototipo.

1.4 Medios empleados

Para la elaboración de este proyecto se ha usado el siguiente software libre:

- Arch Linux x86_64 (kernel 2.6.39)
- VirtualBox 4.0.8 (El servidor de Clouded Desktop corre en una máquina virtual con Ubuntu 10.10 32 bits).

  En el servidor:

  - Apache Server 2.2.16
  - MySQL Server Ver 14.14 Distrib 5.1.49
  - PHP 5.3.3-1ubuntu9.5 con Suhosin-Patch
  - NXServer 3.2.0-74-SVN
- LibreOffice 3.4.0 (para la confección de esta memoria).

1.5 Estructura de esta memoria

En este apartado se comentará de forma breve el contenido de los capítulos del presente documento con el objetivo de facilitar la lectura del mismo:

- Capítulo 1: breve introducción en la que se exponen los objetivos y motivaciones del proyecto, las fases de desarrollo, o los medios empleados para llevarlo a cabo.
- Capítulo 2: comprende un análisis del estado actual de los distintos protocolos de escritorio remoto así como una comparativa entre Clouded Desktop y otro servicio real que funciona en la actualidad.
- Capítulo 3: este capítulo comprende la fase de análisis del ciclo de desarrollo de software y

se explicarán aspectos del diseño y la implementación.

- Capítulo 4: en él se explicarán las conclusiones obtenidas al finalizar el desarrollo así como se indicarán una serie de líneas futuras que podrían ser aplicadas al proyecto.
- Capítulo 5: el último capítulo de este documento contendrá la planificación seguida para el desarrollo del proyecto.
- Referencias y Bibliografía: por último se incluyen las referencias y la bibliografía consultada para la elaboración del proyecto.

2 Estado de la cuestión

2.1 Introducción

En este capítulo se llevará a cabo un pequeño estudio de varios protocolos de escritorio remoto que existen actualmente, profundizando más en la opción elegida para Clouded Desktop. Además, se analizará y comparará un servicio real similar a Clouded Desktop en planteamiento, pero con un funcionamiento interno distinto.

2.2 Estudio de los protocolos de escritorio remoto

Un escritorio remoto no es más que una interfaz gráfica que nos permite controlar una máquina remota como si estuviéramos delante. El concepto no es nuevo y aunque a día de hoy existen muchos productos software que aparentemente sirven para lo mismo, lo cierto es que hay diferencias de rendimiento dependiendo del protocolo de comunicaciones usado para transferir la información entre el cliente y la máquina remota que se pretende controlar. Además del protocolo, existen diferencias como la posibilidad de controlar remotamente aplicaciones individuales en vez de todo el escritorio así como poder conectar desde un navegador web en vez de necesitar una aplicación cliente específica.

2.2.1 RFB

RFB ("remote framebuffer") es un antiguo protocolo de escritorio remoto cuya característica principal es que funciona a nivel de framebuffer representando cada pixel de la pantalla como una zona de memoria. Dicho *grosso modo*, trabaja a "nivel de pixel". Por este motivo es compatible con

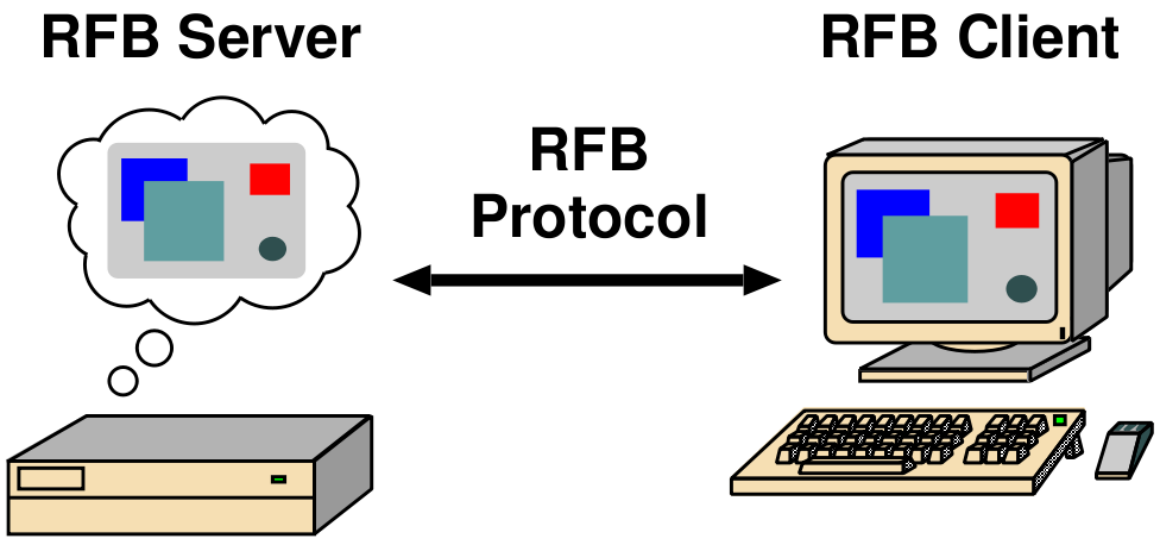

Figura 1: protocolo RFB

cualquier sistema operativo independientemente de su gestor de ventanas.

2.2.2 RDP

Remote Desktop Protocol (RDP) es un protocolo propietario desarrollado por Microsoft que permite la comunicación en la ejecución de una aplicación entre un terminal (mostrando la información procesada que recibe del servidor) y un servidor Windows (recibiendo la información dada por el usuario en el terminal mediante el ratón o el teclado).

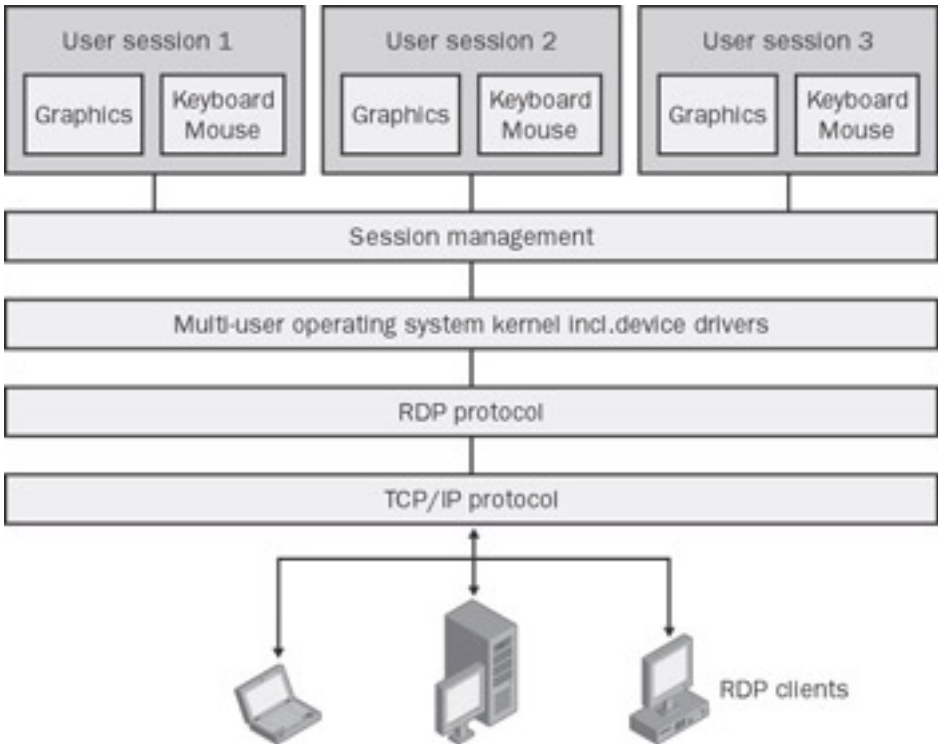

Figura 2: protocolo RDP

El modo de funcionamiento del protocolo no es demasiado complicado. La información gráfica que genera el servidor es convertida a un formato propio RDP y enviada a través de la red al terminal, que interpretará la información contenida en el paquete del protocolo para reconstruir la imagen a mostrar en la pantalla del terminal. En cuanto a la introducción de órdenes en el terminal por parte del usuario, las teclas que pulse el usuario en el teclado del terminal así como los movimientos y pulsaciones de ratón son redirigidos al servidor, permitiendo el protocolo un cifrado de los mismos por motivos de seguridad. El protocolo también permite que toda la información que intercambien cliente y servidor sea comprimida para un mejor rendimiento en las redes menos veloces. Pues es la única de las soluciones de clientes ligeros analizadas que nos permite utilizar este protocolo para que los terminales puedan actuar como clientes de servidores Windows, lo que puede ser interesante en multitud de ambientes de trabajo en los que se utilizan servidores Microsoft.

Este servicio utiliza por defecto el puerto TCP 3389 en el servidor para recibir las peticiones. Una vez iniciada la sesión desde un punto remoto el ordenador servidor mostrará la pantalla de bienvenida de Windows, no se verá lo que el usuario está realizando de forma remota.

Este servicio tiene distintos tipos de aplicaciones: se utiliza frecuentemente para el acceso remoto en la administración de equipos, pero también es cada vez más utilizado en la gestión de servicios de terminal o clientes ligeros (*thin clients*).

Características:

- Permite el uso de colores de 8, 15, 16, 24 y 32 bits
- Cifrado de 128 bits utilizando el algoritmo criptográfico RC4. Los clientes más antiguos pueden utilizar cifrados más débiles.
- Permite seguridad a nivel de transporte Transport Layer Security.
- El redireccionamiento del audio permite al usuario ejecutar un programa de audio en una ventana remota y escuchar el sonido en el ordenador local.
- El redireccionamiento del sistema de ficheros permite a los usuarios utilizar sus ficheros locales en una ventana remota.

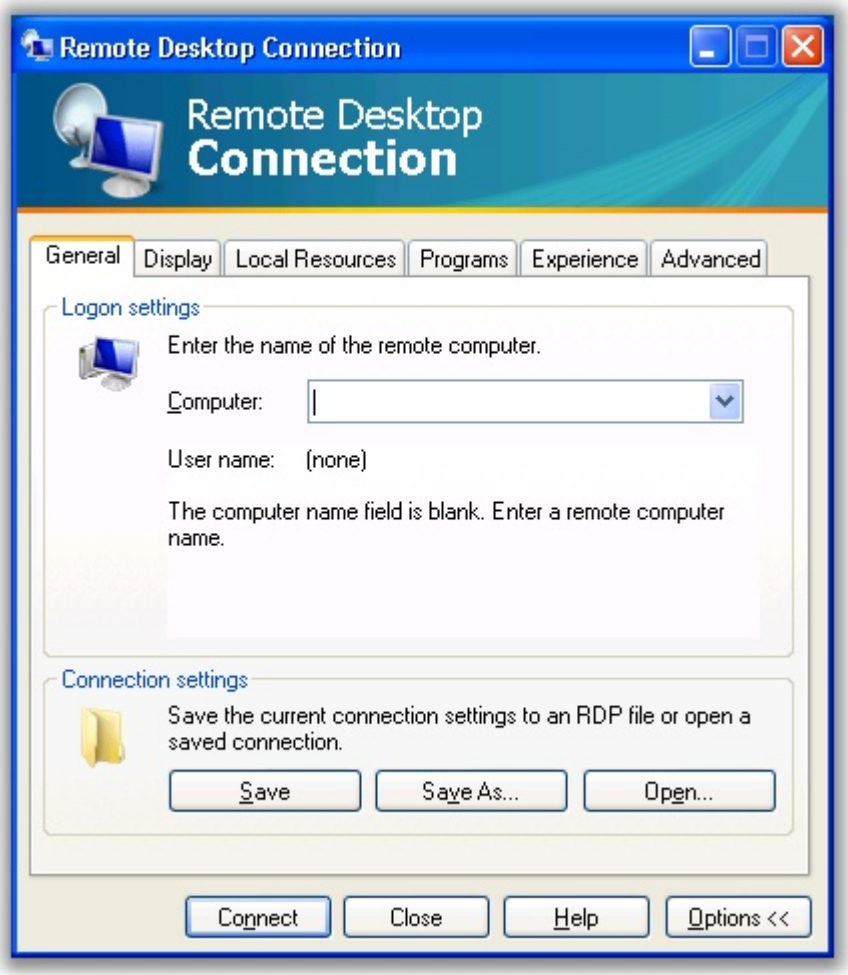

Figura 3: ventana del cliente RDP de Microsoft

- Permite al usuario utilizar su impresora local al estar conectado al sistema remoto.

- El redireccionamiento de puertos permite utilizar los puertos serie y paralelo directamente.
- El portapapeles puede compartirse entre los ordenadores local y remoto.

A partir de 2006 en la versión RDP 6.0 se introdujeron las siguientes características.

- Programas remotos: Aplicaciones con ficheros del lado del cliente.
- Las aplicaciones (seamless Windows) remotas pueden funcionar en una máquina cliente servida por una conexión remota.
- Permite utilizar un servidor IIS de manera que acepte conexiones en el puerto 443 para servidores de respaldo de Terminal Services mediante conexiones HTTPS, similar a como las llamadas remotas RPC sobre HTTP permiten a los clientes Outlook conectar a un servidor de copias Exchange 2003. Se necesita Windows Server 2008.
- Soporte remoto de Aero Glass Thema (o Composed Desktop), incluyendo tecnología de suavizado de fuentes ClearType
- Soporte para aplicaciones Windows Presentation Foundation compatibles con clientes .Net Framework 3.0 y que sean capaces de tener efectos en la máquina local.
- Revisado para que el redireccionamiento de dispositivos sea más general, permitiendo una mayor variedad de dispositivos.
- Todos los servicios de terminal serán totalmente configurables y suscritos vía Windows Management Instrumentation.
- Ancho de banda ajustado para clientes RDP.
- Soporte para Transport Layer Security (TLS) 1.0 en los lados cliente y servidor.
- Soporte de varios monitores. La sesión puede mostrarse en dos monitores.

2.2.3 X Window

X Window System (en español sistema de ventanas X) es un software que fue desarrollado a

mediados de los años 1980 en el MIT para dotar de una interfaz gráfica a los sistemas Unix. Este protocolo permite la interacción gráfica en red entre un usuario y una o más computadoras haciendo transparente la red para éste. Generalmente se refiere a la versión 11 de este protocolo, X11, el que está en uso actualmente. X es el encargado de mostrar la información gráfica de forma totalmente independiente del sistema operativo.

X fue diseñado primariamente para implementar clientes ligeros, donde mucha gente usaba simultáneamente la capacidad de procesamiento de un mismo computador trabajando en tiempo compartido. Cada persona usaba un terminal en red que tenía capacidades limitadas para dibujar la pantalla y aceptar la entrada del usuario. Debido a la ubicuidad del soporte para el software X en Unix, es usado en los computadores personales incluso cuando no hay necesidad del tiempo compartido.

El sistema de ventanas X distribuye el procesamiento de aplicaciones especificando enlaces cliente-servidor. El servidor provee servicios para acceder a la pantalla, teclado y ratón, mientras que los clientes son las aplicaciones que utilizan estos recursos para interacción con el usuario. De este modo mientras el servidor se ejecuta de manera local, las aplicaciones pueden ejecutarse remotamente desde otras máquinas, proporcionando así el concepto de transparencia de red.

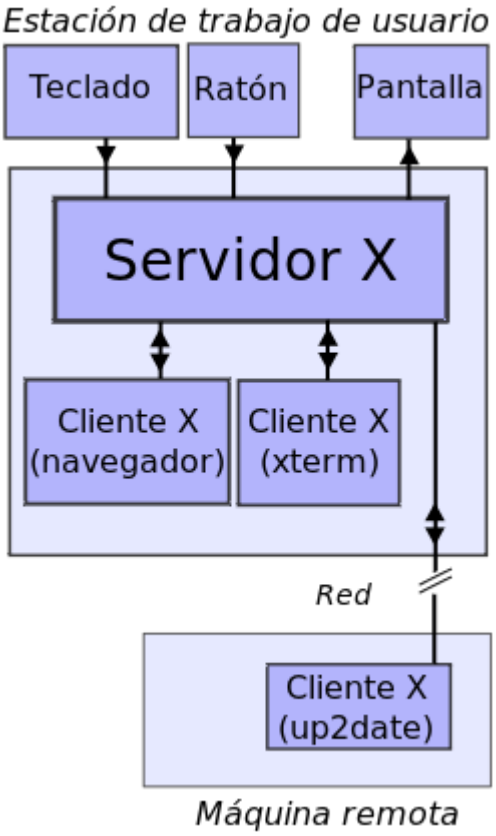

Figura 4: arquitectura de X-Window

Debido a este esquema cliente-servidor, se puede decir que X se comporta como una terminal gráfica virtual.

El hecho que exista un estándar definido para X permite que se desarrollen servidores X para distintos sistemas operativos y plataformas, lo que hace que el código sea muy portable. Por ejemplo. permite tener clientes X ejecutándose en un potente servidor UNIX mientras los resultados son visualizados en una PC de escritorio con cualquier otro sistema operativo funcionando.

La comunicación entre el cliente X y el servidor se realiza por medio de un protocolo conocido como Xprotocol, que consiste en una serie de bytes interpretados como comandos básicos

para generar ventanas, posicionarlas, o controlar eventos. Los clientes X acceden al Xprotocol mediante el uso de una biblioteca llamada Xlib, que evita al programador de clientes X tener que lidiar con el código binario del Xprotocol. Sin embargo, los aspectos de decoración de ventana y manejos de ventanas no están definidos en esta biblioteca.

X no es un gestor de ventanas, necesita de uno para controlar el manejo de ventanas. Esto trae la ventaja de que permite al usuario instalar uno o más administradores de ventanas de su preferencia. También trae la ventaja de que hace de X estrictamente un sistema gráfico, de tal modo que un cliente X podría estar enviando un gráfico a una pantalla, a una impresora o a cualquier otro hardware sin darse cuenta, flexibilizando la salida gráfica.

Por otro lado, la desventaja que trae el hecho de no tener un único entorno gráfico es que los programadores de clientes X que desean hacer uso de los recursos de los entornos gráficos (botones, barras de deslizamientos, etc.) deben elegir un entorno gráfico específico para programar y contar que el usuario tenga por los menos las bibliotecas de dicho entorno gráfico instalado. Las bibliotecas de los entornos gráficos se conocen como "Toolkits", el estándar X provee sólo de un conjunto de herramientas básicas llamadas Xintrisics que permiten a los programadores de los entornos gráficos armar sus Toolkits sobre éstas.

X usa el modelo cliente-servidor: un servidor X se comunica con varios programas cliente. El servidor acepta los pedidos para la salida gráfica (ventanas) y devuelve la entrada del usuario (desde el teclado, del ratón, o de la pantalla táctil). El servidor puede funcionar así:

- una aplicación exhibiendo hacia una ventana de otro sistema de visualización.
- un programa del sistema controlando la salida vídeo de una PC.
- una pieza de hardware dedicada.

Esta terminología de cliente servidor - el terminal de usuario siendo el servidor y las aplicaciones siendo los clientes - a menudo confunde a nuevos usuarios de X, porque los términos parecen invertidos. Pero X toma la perspectiva de la aplicación, en vez de la del usuario final: X proporciona la exhibición por pantalla y los servicios de entrada/salida a las aplicaciones, así que es un servidor; las aplicaciones usan estos servicios, por lo tanto son los clientes.

El protocolo de comunicaciones entre el servidor y el cliente opera transparente a la red: el cliente y el servidor pueden correr en la misma o en diferentes máquinas, posiblemente con diferentes arquitecturas y sistemas operativos. Un cliente y un servidor pueden incluso comunicarse

con seguridad sobre Internet haciendo una conexión de túnel sobre una sesión cifrada de la red.

Un cliente X puede emular un servidor X proporcionando servicios de exhibición a otros clientes. Esto es conocido como "X nesting" (anidado X). Los clientes de código abierto tales como Xnest y Xephyr soportan el X nesting.

Para utilizar un programa de cliente X sobre una máquina remota, el usuario hace lo siguiente:

- En la máquina local, abrir una ventana de terminal
- usar telnet o ssh para conectarse con la máquina remota
- solicitar el servicio local de pantalla/entrada (ej., export DISPLAY=[user's machine]:0 si no se está usando SSH con X tunneling activado)
- El cliente X remoto entonces hará una conexión al servidor X local del usuario, proporcionando la exhibición por pantalla y la entrada.
- Alternativamente, la máquina local puede correr un pequeño programa que se conecte con la máquina remota y comience la aplicación cliente.

Ejemplos prácticos de clientes remotos incluyen:

- administrando una máquina remota gráficamente.
- corriendo una simulación de cómputo intensivo en una máquina Unix remota y mostrando los resultados por pantalla en una de escritorio Windows.
- correr software gráfico en varias máquinas al mismo tiempo, controlados por una sola pantalla, teclado y ratón.

2.2.4 NX technology

NX no es un protocolo propiamente dicho sino una serie de técnicas de compresión y optimización sobre el protocolo X Window que mejoran sustancialmente la transmisión de datos a través de Internet incluso con conexiones catalogadas como “muy lentas”. El principal objetivo de esta tecnología es el de ejecutar a través de Internet aplicaciones que logren tener el mismo aspecto gráfico que cuando se ejecutan en cualquier PC. Generalmente cuando esto ocurre es necesario inhabilitar todo aquello que pueda consumir demasiados recursos, como por ejemplo, menús

desplegables, fondos de pantalla, iconos o animaciones gráficas. NX fue diseñado especialmente para soportar estas condiciones y no hacer que usuarios y desarrolladores deban cambiar sus hábitos o su código.

NX está desarrollado por *Gian Filippo Pinzari* que trabaja para la empresa de software italiana NoMachine. El protocolo está basado en DXPC (Differential X Protocol Compressor Project), hecho público en 1995 por *Brian Pane*, que a parte de servir como fuente de ideas para NX, les proporcionó la codificación diferencial específica de muchas de las peticiones, respuestas y eventos que forman el corazón del protocolo X. Tanto NX como las herramientas que hacen uso de él estás actualmente disponibles para Windows, MacOSX, Linux and Solaris. Hasta la versión 4.0, NoMachine utilizó la licencia GNU para el núcleo de NX al mismo tiempo que ofrecían versiones de pago enfocadas a empresas, un cliente y servidor gratuito para Linux y un cliente gratuito para Microsoft Windows, Mac OS X y sistemas empotrados.

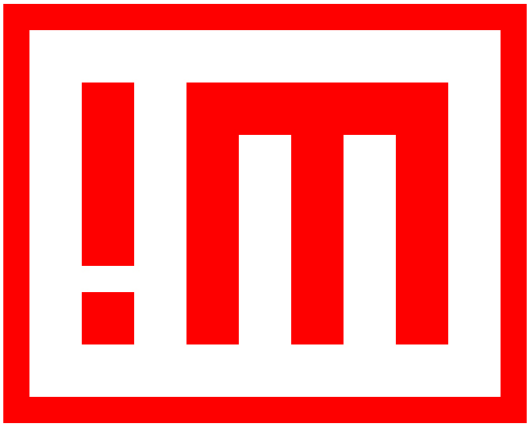

Figura 5: logo de NoMachine

El cliente mayoritariamente usado es el que suministra NoMachine, pero en cuanto al servidor existen otras opciones como por ejemplo *FreeNX* (*Fabian Franz*, GPL) y *Neatx* (*Google* ™, GPL).

2.2.4.1 NX en detalle

Teniendo en cuenta la importancia de NX en mi proyecto, pienso que merece la pena explicar más en profundidad su funcionamiento.

2.2.4.1.1 MessageStore (caché de mensajes X)

NX trata los mensajes del protocolo X como si estuvieran formados por una parte de tamaño fijo, llamada *identidad* y la parte de datos, de tamaño variable. A continuación se muestra la representación en el lenguaje C de una petición *PolySegment* del protocolo X Windows extraído de *Xproto.h*:

```
#define X_PolySegment 66
#define sz_xPolySegmentReq 12
typedef struct{
      CARD8 reqType;
      BYTE pad;
      CARD16 length B16;
      Drawable drawable B32;
      Gcontext gc B32;
}xPolySegmentReq;
```

La *identidad* corresponde normalmente a la estructura de la petición X. En este caso la parte de la *identidad* tiene una longitud de 12 bytes.

NX mantiene en memoria principal una caché de los últimos mensajes X enviados, divididos por el código de operación de protocolo. Esta caché se llama *MessageStore.*

Para permitir una búsqueda rápida de los mensajes en la caché, NX calcula una suma de control MD5 de cualquier petición nueva o respuesta que tiene que ser codificada. Cualquier tipo de mensaje tiene su propio método para calcular el MD5 de la *identidad*, mientras que el hash MD5 de la parte de datos se obtiene simplemente sumando cualquier byte de datos a la suma de control. NX mantiene información acerca del código de operación de forma implícita, añadiendo mensajes en el *MessageStore* adecuado, y ahorrando tamaño en un campo específico del típico básico de mensaje.

Cuando se recibe un mensaje nuevo, NX calcula su suma de control y lo busca en la caché. Si lo encuentra, NX sólo envía su información de estado a la máquina remota, junto con la posición exacta de la caché donde se encuentra el mensaje y una codificación diferencial de todos aquellos campos que no forman parte de la suma de control.

En el caso de *PolySegment,* los campos que necesitan ser codificados de forma diferencial son *Drawable* y *Gcontext.* Todos los valores atómicos que NX envía a través de la red tienen una codificación específica, normalmente basada en una caché de caracteres o enteros.

Codificar la información de estado y la posición del mensaje en la caché requieren entre 2 y 6 bits, dependiendo del protocolo del mensaje. *Drawable* y *Gcontext* (originalmente 64 bits) pueden ser codificados usando entre 2 y 8 bits. Según esto, es posible codificar una petición PolySegment de 176 bits usando sólo de 4 a 14 bits, o lo que es lo mismo, con un ratio de compresión de más de 10:1

En caso de que el mensaje no esté en la caché, el mensaje a enviar es codificado campo a campo y se añade al principio la posición donde de la caché donde debe almacenarlo el equipo remoto. Dado que el lugar de cada mensaje en la caché es controlado por el emisor, puede decirse que tanto servidor como cliente saben en todo momento lo que hay en la caché del otro sin necesidad de preguntar al otro con el ahorro de ancho de banda que eso conlleva.

Es interesante hacer notar que sólo el emisor se encarga de calcular y mantener las sumas de control de los mensajes, y el receptor sólo necesita almacenar los datos en sí. Esto reduce enormemente la cantidad de memoria total requerida para almacenar los más de 3000 mensajes que puede guardar cada caché.

En un principio podría pensarse que los mensajes del protocolo X son demasiado variables como para ser eficientemente cacheados, pero separando la parte variable del cálculo del hash y codificándola por separado, la cantidad de mensajes cacheables aumenta considerablemente. A lo largo de años de mejoras y optimizaciones del algoritmo de codificación, NX ha llegado a conseguir cachear entre un 60 y un 80% del total de mensajes del protocolo X que son enviados. Para otro tipo de mensajes, como peticiones gráficas, imágenes, fuentes, etc., el total de mensajes cacheados puede ser del 100%, permitiendo a NX alcanzar compresiones del orden de 1000:1

Pinzari observó que la codificación basada en la caché de mensajes no sólo reducía el ancho de banda usado por el protocolo X, sino que también obtenía un rendimiento de CPU muy bueno respecto a otros métodos de compresión existentes. El tiempo necesario para calcular el hash MD5 de un mensaje y enviarle la referencia a la máquina remota es mucho más pequeño que el tiempo empleado para comprimir el mensaje X con un algoritmo de compresión genérico como por ejemplo ZLIB.

2.2.4.1.2 Codificación diferencial del protocolo X

Como ya se comentó en el punto anterior, cuando NX no encuentra un mensaje similar en la caché utiliza lo que se denomina codificación diferencial. En ese caso, el mensaje es codificado campo a campo, pero teniendo cuidado de que los bytes de relleno así como la información implícita no sean enviados.

Para minimizar el número de bits requeridos para codificar el mensaje, todas los mensajes más comunes tienen su propio método de codificación diferencial. En el caso de *PolySegment*, por ejemplo, cualquier coordenada es enviada como un entero con signo que representa la diferencia

respecto a la coordenada X ó Y previa. Normalmente es posible codificar esa diferencia con menos de 8 bits, en vez de los 16 bits requeridos por la codificación original usada por el protocolo X. Además de esto, cada campo del mensaje tiene su propia caché de enteros, por lo que dicha diferencia es buscada antes de ser enviada en la caché. De media, un mensaje de tipo *PolySegment* que requiere 32 bytes puede ser reducido como máximo a 32 bits, y de forma general, la codificación diferencial permite ratios de compresión que van de 5:1 hasta 8:1.

2.2.4.1.3 ZLIB

Las partes de datos de los mensajes que no disponen de un método específico de codificación diferencial, son comprimidas utilizando ZLIB. Esta compresión puede ser bastante efectiva alcanzando de media ratios de 4:1. De todas formas, ZLIB es usado en raras ocasiones como por ejemplo con imágenes en blanco y negro para las cuales cualquier otro tipo de compresión resultaría peor o para respuestas de tipo *GetProperty,* para las que no es posible determinar el mejor método de codificación diferencial. Los datos comprimidos con ZLIB, se guardan en caché también comprimidos ya que el ahorro de CPU es despreciable frente al ahorro de espacio en la caché.

2.2.4.1.4 Mini caché para campos int y char

Como se comentó antes, los campos enteros más comunes de las peticiones tienen su propia caché. Existen dos tipos de valores en esta caché. Los enteros de 32 bits y los caracteres, para valores de hasta 8 bits. Estas cachés utilizan un algoritmo simple de lista circular. Por ejemplo, ya que los clientes X suelen usar el mismo ID de ventana en múltiples mensajes, usando esta mini caché se consigue reducir los 32 bits del ID a 1 bit.

2.2.4.1.5 Mini caché para campos compuestos

A continuación se muestra el código C++ de una caché típica de un valor compuesto:

```
enum T_status{
      is_added,
      is_cached,
      is_discarded,
      is_removed
};
class StatusCache{
      friend class EncodeBuffer;
      friend class DecodeBuffer;
      public:
            StatusCache(){
                  slot_=0;
            }
            ~StatusCache(){}

      private:
            CharCache base_[4];
            unsigned char slot_;
};
```

Esta caché es usada para comunicarle a la máquina remota qué hacer con el siguiente mensaje X que va a recibir:

```
void encodeStatusValue(unsigned char next, StatusCache &cache){
      encodeCachedValue(next, 2, cache.base_[cache.slot_]);
      cache.slot_=next;
}
```

El estado es representado por, como máximo, 2 bits, pero cualquier valor nuevo es codificado basándose en el previo usando un algoritmo de lista circular. Si el valor es encontrado en la primera posición de la caché asociada al slot actual, sólo se necesitará 1 bit para codificarlo.

Un método similar es usado para codificar *GCs*, *Drawables* y otros valores XIDs característicos del protocolo X. Aprovechando la recurrencia de esos valores especiales de 32 bits en los distintos mensajes enviados por el protocolo X se pueden alcanzar unos ratios de compresión muy buenos. Las peticiones *ChangeGCs*, que necesitan de media entre 16 y 18 bytes, son codificadas normalmente con 16 bits, con el valor *GC* codificado así mismo usando entre 1 y 8 bits.

2.2.4.1.6 Compresión de imágenes

NX introduce un mensaje especial en el protocolo X, *X_PutPackedImage*, para el tratamiento de imágenes. La compresión de las imágenes es llevada a cabo por NX cada vez que se necesita enviar un mensaje *X_PutImage* al servidor X. NX consulta a la máquina remota al iniciar la sesión a fin de acordar qué métodos de compresión se usarán durante la sesión. El método elegido normalmente dependerá del ancho de banda disponible. Las imágenes comprimidas se almacenan así y son descomprimidas al vuelo en la máquina remota siempre que un mensaje *X_PutImage* tiene que ser enviado al servidor X.

La eficiencia de la compresión de imágenes de NX es una combinación del algoritmo de compresión utilizado (por ejemplo JPEG, PNG, RDP, TIGHT, ZLIB) así como del uso de la caché y la codificación diferencial. Como dato, con una imagen típica JPEG se puede conseguir un ratio de compresión 20:1. Si además de eso añadimos codificación diferencial con un ratio de 100:1, tenemos que en total se puede conseguir una compresión de 2000:1

2.2.4.1.7 Streaming de imágenes

Los mejores métodos de compresión de imágenes no son suficientes para permitir una interacción fluida del usuario en conexiones “muy lentas”. Por ejemplo, la transferencia completa de una imagen a través del protocolo X que tuviera un tamaño original de 256 KB, comprimida en JPEG con un ratio de 20:1 hasta los 12 KB, requeriría 4 segundos para ser enviada a través de un módem antes de que cualquier aplicación fuera capaz de mostrar algo por pantalla.

NX trocea los mensajes grandes (como las imágenes) en *chunks* que envía sólo cuando ningún otro mensaje del protocolo X necesita ser enviado. El algoritmo utilizado penaliza a los clientes X que hacen un uso intensivo de peticiones de imágenes para no saturar al resto. Por otro lado NX implementa un algoritmo de reparto de tiempo entre todas las aplicaciones que envían mensajes al servidor X a fin de no dejar bloqueadas otras aplicaciones cuando alguna petición grande es enviada. Básicamente, cuando esto ocurre, NX “duerme” la aplicación hasta que la petición es procesada en la máquina remota.

El método por defecto actual para codificar imágenes en las sesiones NX es JPEG, con una calidad determinada en función del ancho de banda de la conexión. Otros métodos disponibles son PNG y ZLIB.

2.2.4.1.8 Control del ancho de banda

¿Qué pasaría si un usuario intentara ejecutar un salva-pantallas a través de una sesión X remota? El salva-pantallas consumiría todo el ancho de banda de subida disponible, bloqueando a otros usuarios conectados a la máquina. Para evitar este tipo de situaciones, NX asigna a cada "aplicación" de la máquina local una cuota de ancho de banda. Cuando esta cuota es superada, NX duerme a la aplicación hasta que la conexión esté disponible.

Para reducir el retardo entre las acciones de los usuarios y el envío de los mensajes, NX mantiene una cantidad de datos prefijada en la ventana TCP dependiendo del ancho de banda de la conexión. Para conseguir esto, el buffer TCP es monitorizado y sólo son aceptados los bytes necesarios para completar un paquete TCP del tamaño prefijado.

2.2.4.1.9 Encapsulamiento y compresión de los protocolos RDP y RFB

NX es capaz de encapsular y comprimir otros protocolos de escritorio remoto como son RFB y RDP. Para ello, las actualizaciones de pantalla son transformadas en peticiones del protocolo X, como por ejemplo *X_PolyFillRectangle* o *X_ClearArea*. Cuando las actualizaciones de pantalla son codificadas como imágenes, NX envía estas imágenes en su formato original, aplicando la usual codificación diferencial y algoritmos de cacheado. Las imágenes de RDP o RFB son codificadas a través de peticiones *NX_PutPackedImage* usando métodos de empaquetado como PACK_RDP_COMPRESSED_256_COLORS, PACK_RFB_HEXTILE, PACK_RFB_TIGHT_PLAIN, etc. La máquina remota sabe como transformar estas imágenes especiales en peticiones *X_PutImage*.

2.2.4.1.10 Compresión de la trama final

Antes de enviar el paquete, NX vuelve a comprimir la trama entera usando ZLIB con una grado de compresión en función del ancho de banda y latencia de la conexión. Este paso final puede ahorrar hasta un 30% de ancho de banda a costa de sacrificar muy poco tiempo de CPU.

2.2.4.1.11 Persistencia de las cachés de mensajes

Cada vez que se termina la sesión, NX guarda todas las cachés en disco, calculando una suma de control de cada una tanto en la máquina local como en la remota. De esta forma se

consigue evitar la recarga en memoria de cachés corruptas.

2.2.4.1.12 Persistencia de la caché de imágenes

La persistencia de las imágenes en disco funciona junto a la caché de mensajes siempre que las imágenes no se encuentren en la memoria principal. La máquina local almacena en disco las imágenes que pasan de cierto tamaño etiquetándolas según los primeros dígitos de su suma de control. Cuando empieza a recibir una imagen de una máquina remota (la cual primero le envía la suma de control de la imagen completa), la maquina local busca en la caché y si encuentra la imagen envía un mensaje asíncrono a la máquina remota para abortar la transferencia. Al ser un mensaje asíncrono puede que llegue después de terminar la transferencia o no. En caso de llegar antes, la máquina remota abortará la transmisión. El algoritmo está pensando teniendo en cuenta que no todas las imágenes tienen por qué ser cacheadas en disco, dado que el tiempo que se tarda en recibir un mensaje de cancelación es a menudo mayor que el de la transferencia de la imagen completa.

2.2.5 Comparativa entre los tres:

	RFB	RDP	NX
Cliente multiplataforma	✓	✓	✓
Servidor multiplataforma	✓	✗ (Sólo Windows)	✗ (Sólo GNU/Linux)
Cliente Java	✓	✓	✓
Cifrado integrado	✓	✓	✓
Modo "Seamless"	✗	✓	✓
Múltiples sesiones	✗	✓	✓

Tabla 1: comparativa entre RFB, RDP y NX

2.2.5.1 ¿Por qué NX?

RFB quedó descartado por las siguiente razones:

- Ausencia de multisesión: no es posible que más de un usuario conecte al mismo tiempo al servidor.
- Carecer de modo "seamless" (en esta modalidad el servidor envía al cliente sólo la ventana de la aplicación en ejecución en vez del escritorio remoto entero).
- Baja eficiencia: al funcionar a nivel de framebuffer y carecer de compresión y/o otras optimizaciones resulta demasiado lento al transmitir a través de Internet.

Por tanto la duda estaba entre RDP o NX. Dado que uno de los objetivos del proyecto era ejecutar aplicaciones multiplataforma, la duda de usar un servidor Windows y el protocolo RDP frente a un servidor GNU/Linux y NX quedaba resuelta a favor de lo segundo ya que en GNU/Linux con Wine se pueden ejecutar aplicaciones de Windows pero para ejecutar aplicaciones de GNU/Linux desde Windows a día de hoy es necesario instalar software de virtualización como VirtualBox, crear una máquina virtual, instalar una distribución de GNU/Linux y dentro de ésta instalar la aplicación elegida, lo cual suponía una carga excesiva para el servidor. Además de este aspecto, el sistema de permisos de los sistemas tipo Unix es muy potente y encajaba perfectamente con las necesidades del proyecto.

2.3 Una alternativa real: Spoon

Figura 6: Spoon

Spoon es un servicio web desde el cual se pueden ejecutar aplicaciones de escritorio de todo tipo sin necesidad de instalarlas físicamente en nuestro equipo. El planteamiento general es el mismo que el de mi proyecto, si embargo su diseño y funcionamiento interno difieren.

2.3.1 Funcionamiento

Para poder cargar las aplicaciones de Spoon es necesario descargar e instalar un plugin para el navegador web. A la fecha de redacción de esta memoria, Spoon permite cargar aplicaciones (de 32 y 64 bits) para Microsoft Windows en estos navegadores (todos en sus versiones para Windows): Internet Explorer 6+, Firefox 3+, Safari 2+, Opera 9+ y Chrome.

El principio de funcionamiento de Spoon está basado en dos pilares: virtualización y streaming. Las aplicaciones son transformadas (utilizando una herramienta llamada Spoon Studio) en imágenes virtuales que son descargadas y ejecutadas en nuestro ordenador de forma transparente mediante una pequeña máquina virtual contenida en el plugin del navegador. Esas imágenes virtuales incluyen todos los ficheros, datos de registro, componentes, librerías, etc. necesarios para la carga de la aplicación. La ventaja principal que presentan frente a ficheros ejecutables normales es que pueden empezar a ejecutarse con un 10% de su tamaño descargado y dejar la transferencia del resto del código ejecutable en segundo plano para cuando realmente sea necesario, lo que acelera enormemente la carga de la aplicación.

2.3.1.1 Máquina virtual de Spoon

A diferencia de otras soluciones de virtualización que requieren una copia completa del sistema operativo anfitrión, la máquina virtual de Spoon emula sólo las características requeridas del sistema anfitrión requeridas para la ejecución de una aplicación concreta. Como resultado, las aplicaciones virtualizadas de Spoon tienen casi el mismo rendimiento que las aplicaciones nativas, pero sin provocar cambios en la infraestructura del sistema.

El núcleo de la tecnología de virtualización de Spoon es una pequeña máquina virtual de menos de 1MB que implementa de forma ligera las APIs del sistema operativo anfitrión incluyendo el sistema de ficheros, registro, procesos e hilos. Todo esto corre en modo usuario, lo que significa que no es necesario instalar ningún driver ni hacer usos de permisos especiales de ejecución. Las aplicaciones virtualizadas interactúan con un sistema de ficheros, registro y servicios virtualizados en vez de acceder directamente a los del sistema operativo anfitrión.

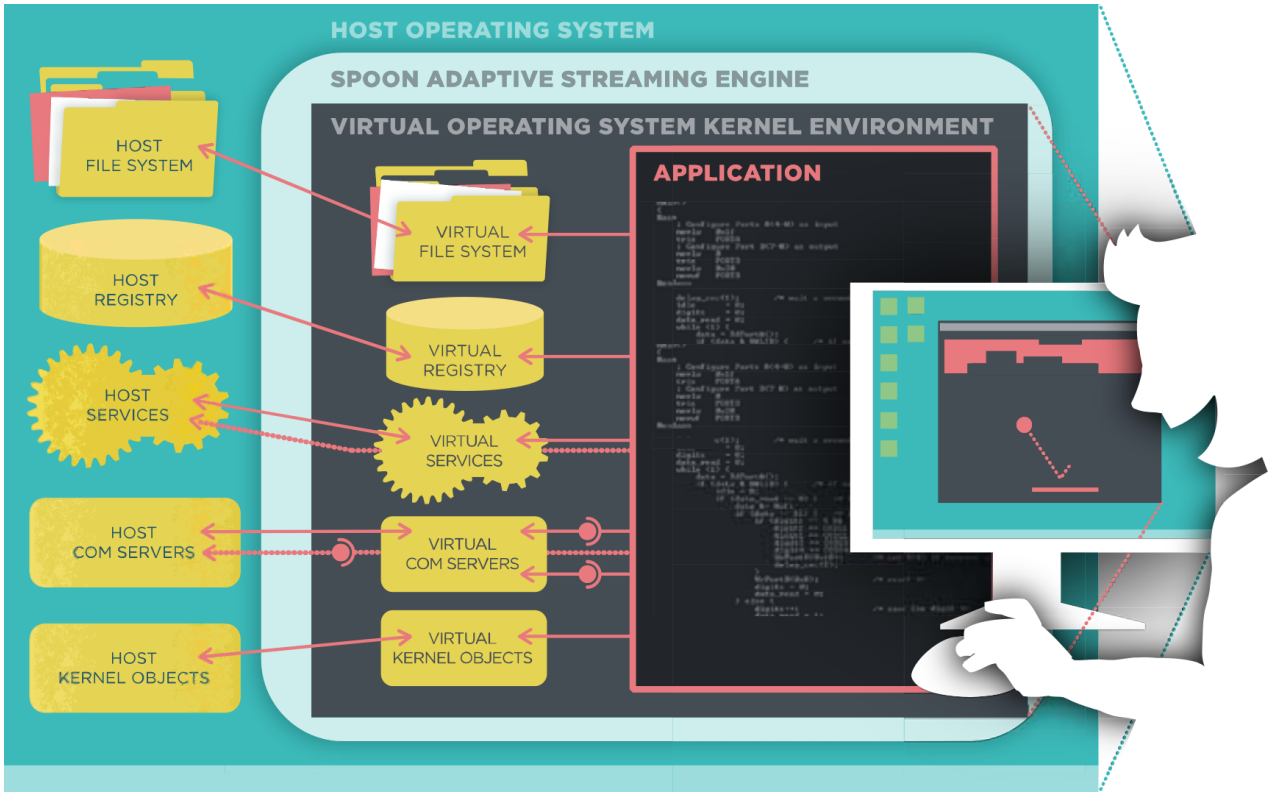

Figura 7: esquema de la máquina virtual de Spoon

2.3.1.2 Spoon Streaming

La exclusiva tecnología de predicción de Spoon streaming permite que las aplicaciones virtuales carguen de 5 a 20 veces más rápido que las aplicaciones descargables. Básicamente Spoon descompone de forma automática las aplicaciones virtuales en pequeñas unidades funcionales y de datos. Después identifica un *prefetch* (los datos vitales para poder cargar la aplicación) y los transfiere primero, lo que permite lanzar la aplicación con sólo una fracción de su tamaño descargado. Una vez que el *prefetch* se transfiere, la aplicación se inicia de inmediato, sin necesidad de ningún tipo de servidor de streaming o protocolo especializado. Las aplicaciones opcionalmente se pueden almacenar en un dispositivo de almacenamiento local al finalizar la transferencia, para acelerar aún más posteriores usos.

2.3.2 Review

A continuación se muestra un pequeño análisis de Spoon a la fecha de la redacción de este apartado.

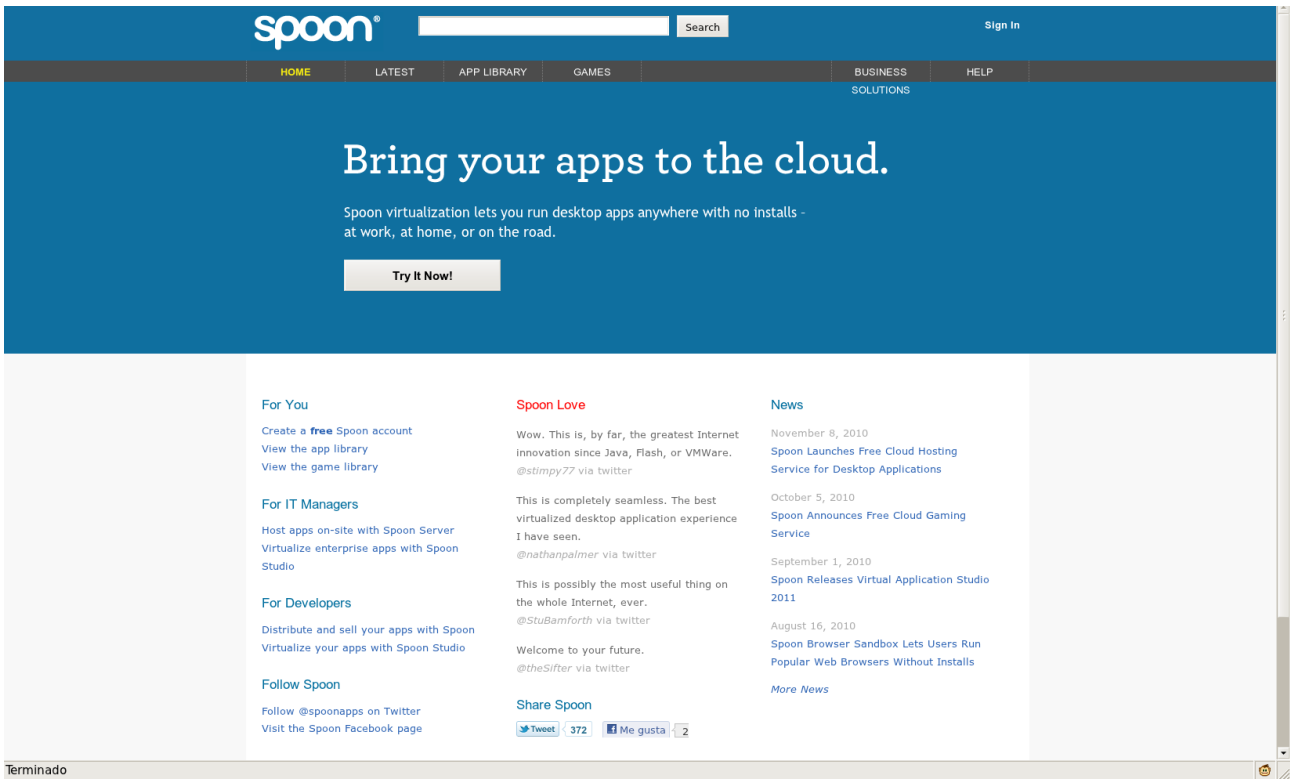

Figura 8: Spoon (página principal)

Como puede apreciarse en las capturas la interfaz es de tipo minimalista. En la parte de arriba hay un buscador simple para localizar aplicaciones por su nombre. Debajo encontramos un menú sencillo con cuatro opciones: un enlace a la página principal, una lista de las últimas aplicaciones subidas a su servidor, un listado completo de todas las aplicaciones disponibles ordenadas por nombre y/o categoría y/o nota de los usuarios y por último un apartado específico para juegos.

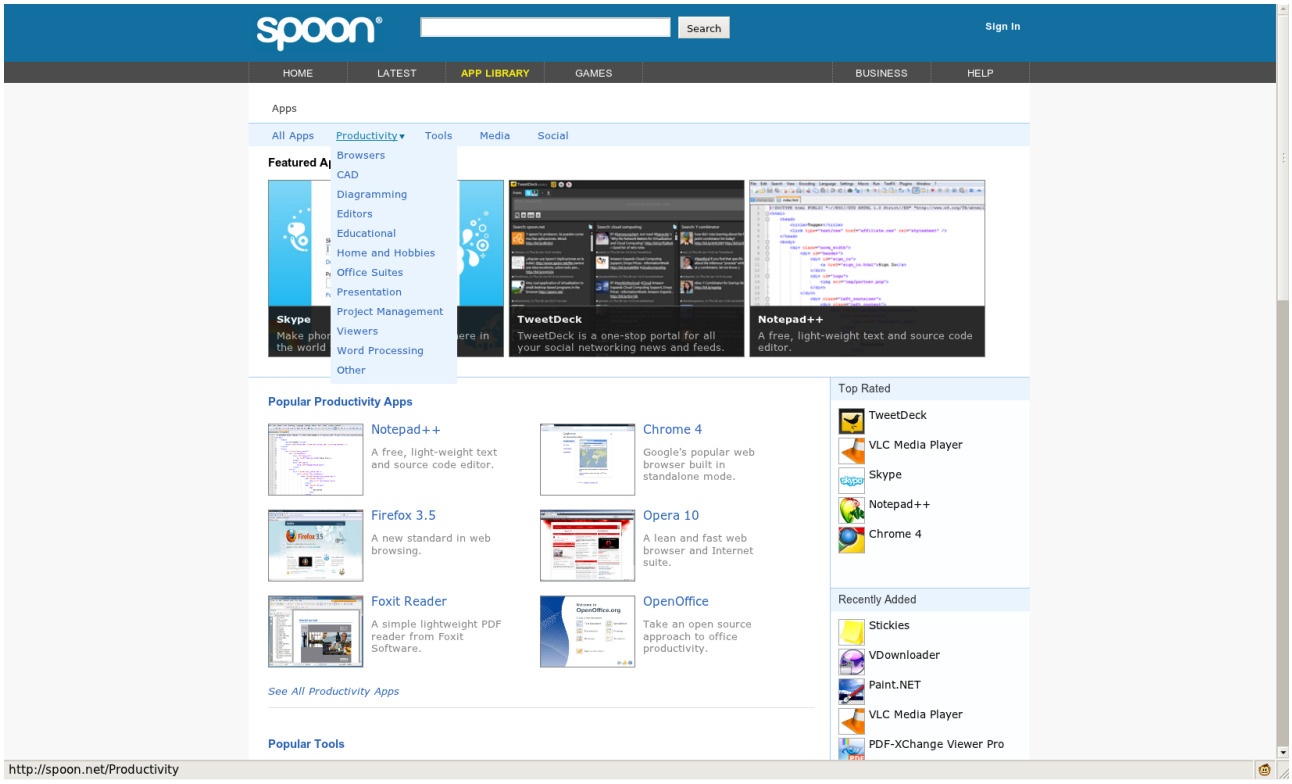

Figura 9: Spoon (librería de aplicaciones)

Figura 10: Spoon (VLC)

Una vez elegida la aplicación, si pinchamos en su nombre se nos cargará una pantalla con información sobre ella (figura 10), a saber: tamaño del ejecutable virtualizado, versión, fecha de publicación y un enlace a la web oficial. También se muestra un pequeño vídeo de funcionamiento y un listado de aplicaciones parecidas. Para cargar la aplicación bastará con pulsar sobre el botón **[LAUNCH NOW]** (Nota: para poder cargar aplicaciones es necesario estar registrado y haber iniciado sesión previamente en la web. El registro es gratuito y sólo se necesita una dirección de email válida y una contraseña).

En la figura 11 puede verse cómo la aplicación VLC es cargada como si fuera una aplicación nativa del sistema. Además de este proceso, en el administrador de tareas figuran dos procesos asociados a la máquina virtual de Spoon *Spoon-Sandbox.exe* y *Spoon-Sync.exe*

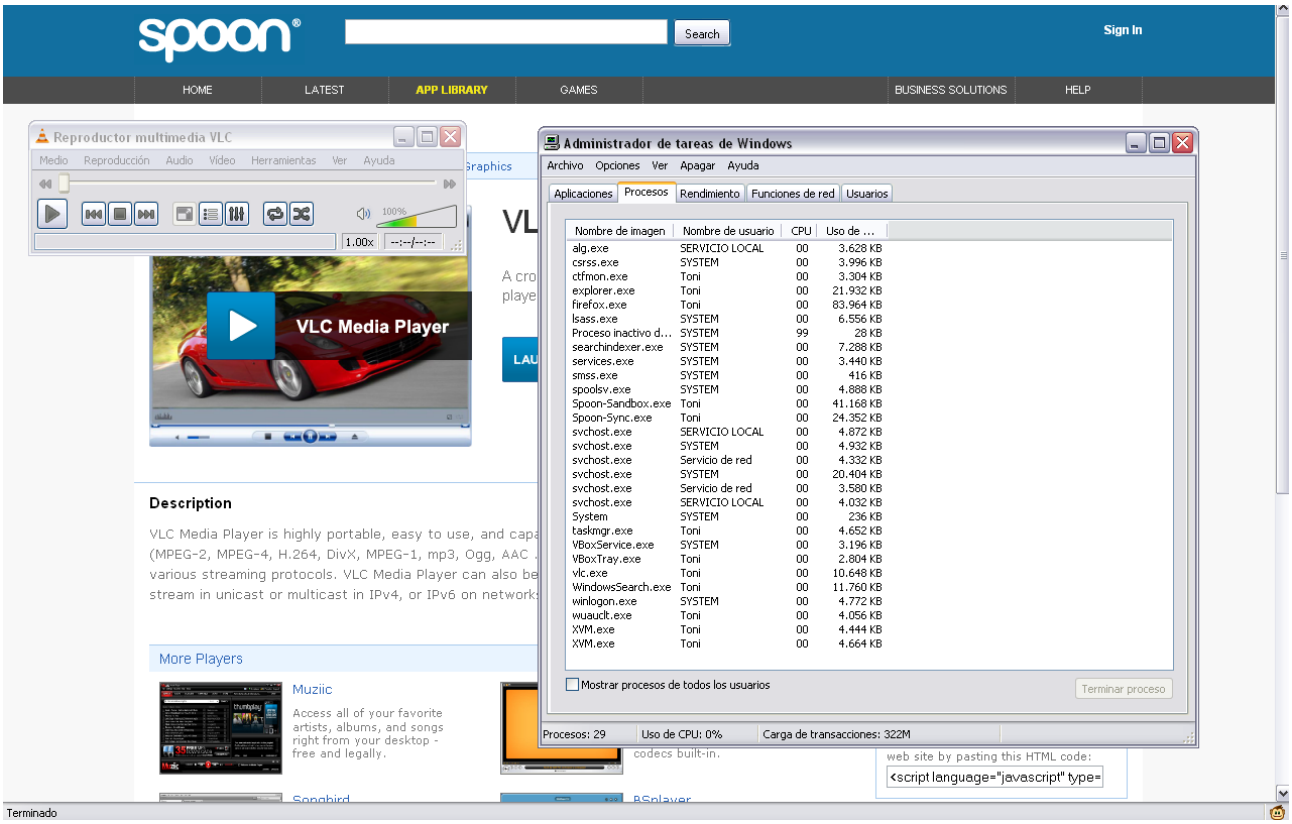

Figura 11: Spoon VLC cargado

2.4 Clouded Desktop vs Spoon

Características	Clouded Desktop	Spoon
Interfaz web	✓	✓
Requiere plugin para el navegador web	✓ Java JRE	✓ Spoon Plugin
Multiplataforma	✓	✗ Sólo Microsoft Windows.
Soporte para aplicaciones de distintas plataformas	✓	✗ Sólo aplicaciones de Microsoft Windows.
Soporte de aplicaciones de 32 y 64 bits	✓	✓
Ejecución “seamless” de las aplicaciones	✓	✓
Las aplicaciones se instalan normalmente en el servidor.	✓	✗ Las aplicaciones tienen que ser previamente “virtualizadas” con Spoon Studio.
Soporte completo de aplicaciones de vídeo	✗ Lo tendrá a partir de la versión 4.0 de NXServer.	✓
Sincronización automática de ficheros en “la nube”	✓ Usando Dropbox de forma opcional.	✗ Los ficheros se guardan localmente.

Tabla 2: comparativa entre Clouded Desktop y Spoon

3 Análisis, diseño e implementación de Clouded Desktop

3.1 Introducción

En primer lugar se realizará un análisis de los requisitos de usuario de la aplicación web, estableciendo además varios casos de uso de la misma. A continuación se abordará la fase de diseño del proyecto. Y por último se detallarán los aspectos más significativos de la implementación de Clouded Desktop.

3.2 Análisis

En este apartado se van a describir los distintos requisitos de usuario de la aplicación clasificándolos como requisitos de capacidad o de restricción. Cada requisito está especificado siguiendo un formato tabular conteniendo la siguiente información para cada uno de ellos:

- Identificador: identifica de forma univoca un requisito, siguiendo las reglas de nombrado siguientes: CAP_#, para los requisitos de capacidad, y RES_#, para los requisitos de restricción. El valor de # es numérico empezando en uno.
- Descripción: especificación detallada del requisito.
- Necesidad: establece la necesidad del requisito dentro del proyecto, toma un valor dentro de la escala 1-4, siendo 4 la necesidad más alta y 1 la más baja.
- Prioridad: establece la prioridad del requisito dentro del proyecto, toma un valor dentro de la escala 1-4, siendo 4 la prioridad más alta y 1 la más baja.

3.2.1 Requisitos de capacidad

A continuación se muestran los requisitos de usuario de capacidad:

Identificador	CAP_1: listar aplicaciones disponibles.
Descripción	El usuario será capaz de ver un listado con todas las aplicaciones disponibles.
Necesidad	4
Prioridad	4

Tabla 3: requisito de capacidad (listar aplicaciones)

Identificador	CAP_2: mostrar aplicaciones agrupadas por categoría.
Descripción	Las aplicaciones se mostrarán agrupadas según la categoría a la que pertenezcan.
Necesidad	4
Prioridad	4

Tabla 4: requisito de capacidad (agrupar aplicaciones)

Identificador	CAP_3: mostrar categorías y aplicaciones alfabéticamente.
Descripción	Las categorías y las aplicaciones se mostrarán ordenadas alfabéticamente.
Necesidad	4
Prioridad	4

Tabla 5: requisito de capacidad (mostrar categorías y aplicaciones alfabéticamente)

Identificador	CAP_4: mostrar descripción de cada aplicación.
Descripción	Las aplicaciones mostrarán una pequeña descripción.
Necesidad	4
Prioridad	4

Tabla 6: requisito de capacidad (mostrar descripción aplicación)

identificador	CAP_5: ejecución paralela de varias aplicaciones.
Descripción	El usuario podrá ejecutar al mismo tiempo varias aplicaciones.
Necesidad	4
Prioridad	4

Tabla 7: requisito de capacidad (ejecución paralela de aplicaciones)

Identificador	CAP_6: persistencia en “la nube” de los ficheros creados.
Descripción	El trabajo realizado con cada aplicación podrá ser guardado en “la nube” para futuras sesiones.
Necesidad	4
Prioridad	4

Tabla 8: requisito de capacidad (persistencia de ficheros en “la nube”)

Identificador	CAP_7: utilizar ficheros externos por las aplicaciones.
Descripción	Las aplicaciones podrán operar con ficheros que no hayan sido creados previamente desde Clouded Desktop siempre que sean subidos a “la nube”.
Necesidad	4
Prioridad	4

Tabla 9: requisito de capacidad (utilizar ficheros externos)

Identificador	CAP_8: descargar ficheros creados.
Descripción	El usuario podrá descargar en cualquier momento los ficheros creados con cualquier aplicación a un soporte de almacenamiento local.
Necesidad	4
Prioridad	4

Tabla 10: requisito de capacidad (descargar ficheros creados)

Identificador	CAP_9: crear lista de aplicaciones favoritas.
Descripción	El usuario podrá crear una lista ordenable de aplicaciones favoritas para acceder más rápido en futuras sesiones.
Necesidad	4
Prioridad	4

Tabla 11: requisito de capacidad (lista aplicaciones favoritas)

Identificador	CAP_10: ejecución “seamless” de las aplicaciones.
Descripción	Las aplicaciones se ejecutarán aparentemente como si fueran aplicaciones “nativas” del equipo, mostrándose en la barra de tareas y siendo posible maximizar o minimizar su ventana.
Necesidad	4
Prioridad	4

Tabla 12: requisito de capacidad (ejecución “seamless”)

Identificador	CAP_11: “portapapeles” compartido.
Descripción	Las aplicaciones de Clouded Desktop y las “nativas” compartirán el “portapapeles” pudiendo copiar y pegar trozos de texto de forma bidireccional entre ellas.
Necesidad	4
Prioridad	4

Tabla 13: requisito de capacidad (“portapapeles” compartido)

3.2.2 Requisitos de restricción

3.2.2.1 Usuario

A continuación se exponen los requisitos de restricción de usuario:

Identificador	RES_1: sistema operativo.

Descripción	• Windows 2000/2003/XP/Vista/7
	• Linux (i386 y x86_64) Comprobado en:
	  • Arch Linux
	  • Red Hat Enterprise Linux 4/5/6
	  • SuSe 10/10.1/10.2/10.3/11/Enterprise 10
	  • Mandriva 110.1/2005/2006/2007/2008/2009/2010
	  • Fedora Linux Core 3/4/5/6
	  • Fedora 7/8/9/10/11/12/13/14
	  • Debian GNU/Linux 4.0 Etch/5.0 Lenny/6.0 Squeeze
	  • Ubuntu 5.10 Breezy Badger/6.06 Dapper Drake/6.10 Edgy Eft/7.04 Feisty Fawn/7.10 Gutsy Gibbon/8.04 Hardy Heron/8.10 Intrepid Ibex/9.04 Jaunty Jackalope/9.10 Karmic Koala/10.4 Lucid Lynx/10.10 Maverick Meerkat/11.04 Natty Narwhal
	  • Xandros Desktop 4.1/Xandros Server 2.0
	• Mac OS X PPC/i386
	  • 10.3-10.6
	• Solaris SPARC
	  • 8/9/10

Necesidad	4
Prioridad	4

Tabla 14: requisito de restricción (sistema operativo)

Identificador	RES_2: navegador web con soporte JRE 1.2 o posterior.
Descripción	Clouded Desktop necesita para funcionar un navegador con soporte para Java.
Necesidad	4
Prioridad	4

Tabla 15: requisito de restricción (Java)

Identificador	RES_3: navegador web con soporte Javascript.
Descripción	Clouded Desktop necesita para funcionar un navegador compatible con Javascript.
Necesidad	4
Prioridad	4

Tabla 16: requisito de restricción (Javascript)

Identificador	RES_4: acceso protegido a la aplicación.
Descripción	El usuario deberá identificarse con un nombre de usuario y contraseña para acceder a Clouded Desktop.
Necesidad	4
Prioridad	4

Tabla 17: requisito de restricción (login)

Identificador	RES_5: cuenta de Dropbox.
Descripción	Clouded Desktop utiliza Dropbox como sistema de apoyo de almacenamiento en "la nube". Si el usuario quiere disfrutar de esta característica deberá crearse previamente una cuenta en Dropbox que utilizará desde Clouded Desktop.
Necesidad	4
Prioridad	4

Tabla 18: requisito de restricción (Dropbox)

3.2.2.2 Servidor

Aquí se muestran los requisitos de la plataforma donde se instale la parte "servidora" de la aplicación:

Identificador	RES_6: sistema operativo.
Descripción	• GNU/Linux (i386 x86_64) (prototipo probado en Ubuntu 10.10 Maverick Kernel 2.6.35-28-generic)
Necesidad	4
Prioridad	4

Tabla 19: requisito de restricción (sistema operativo)

Identificador	RES_7: servidor HTTP.
Descripción	Apache HTTP Server (prototipo probado con versión 2.2.16)
Necesidad	4
Prioridad	4

Tabla 20: requisito de restricción (servidor HTTP)

Identificador	RES_8: PHP 5.
Descripción	Módulo PHP para Apache e intérprete de línea de órdenes. (prototipo probado con versión 5.3.3-1ubuntu9.3)
Necesidad	4
Prioridad	4

Tabla 21: requisito de restricción (PHP)

Identificador	RES_9: MYSQL.
Descripción	MYSQL server (prototipo probado con versión 5.1.49-1ubuntu8.1)
Necesidad	4
Prioridad	4

Tabla 22: requisito de restricción (MySQL)

3.2.3 Casos de uso

Durante este apartado se especificarán los diferentes casos de uso planteados para la aplicación que va a ser desarrollada.

Cada caso de uso va a ser especificado de forma gráfica con un diagrama que muestre la interacción del usuario con el sistema, y además se detallará los pasos necesarios para cada escenario con un formato tabular descrito a continuación:

- Identificador: identifica de forma univoca un caso de uso siguiendo el formato de nombrado CU_X, donde X se corresponde a un número empezando en la unidad.
- Descripción: describe los pasos realizados por el usuario para la situación planteada.
- Precondiciones: condiciones que deben darse para la realización del caso de uso.

- Postcondiciones: condiciones que son resultado de la ejecución del caso de uso.

Identificador	CU_1: iniciar sesión en Clouded Desktop.
Descripción	• El usuario abre su navegador web. • El usuario introduce la URL de Clouded Desktop. • El usuario introduce su nick y contraseña y pulsa el botón “Iniciar sesión”.
Precondiciones	• El usuario ha sido registrado en el sistema previamente por un administrador. • El usuario no ha iniciado sesión en el sistema.
Postcondiciones	• El usuario ha iniciado sesión en el sistema.

Tabla 23: caso de uso (iniciar sesión)

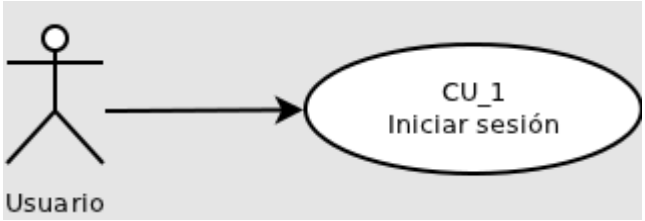

Figura 12: caso de uso (iniciar sesión)

Identificador	CU_2: cerrar sesión en Clouded Desktop.
Descripción	• El usuario pulsa sobre el link “cerrar sesión”
Precondiciones	• El usuario ha iniciado sesión previamente en el sistema.
Postcondiciones	• El usuario ha cerrado sesión en el sistema.

Tabla 24: caso de uso (cerrar sesión)

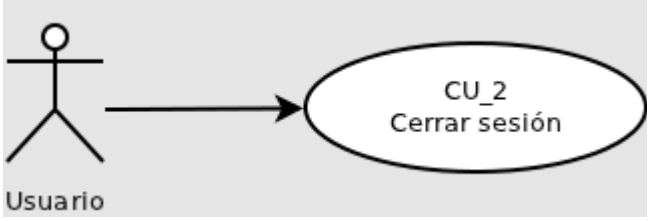

Figura 13: caso de uso (cerrar sesión)

Identificador	CU_3: cargar aplicación.
Descripción	• El usuario elige una categoría en el menú de aplicaciones y dentro de la categoría hace click sobre la aplicación deseada. • Después de hacer click sobre el nombre de la aplicación, se mostrará una descripción de dicha aplicación. • Al pulsar en el enlace [CARGAR APLICACIÓN], se abrirá una ventana modal con el applet de NOMACHINE. • El usuario pulsará en el botón CONTINUE para comenzar la conexión con el servidor. • Después medio minuto aproximadamente, la aplicación elegida se cargará. (El tiempo de carga dependerá de la carga de CPU actual del servidor y de la velocidad de la conexión).
Precondiciones	• El usuario ha iniciado sesión previamente en el sistema. • La aplicación no está en ejecución.
Postcondiciones	• La aplicación se ha cargado.

Tabla 25: caso de uso (cargar aplicación)

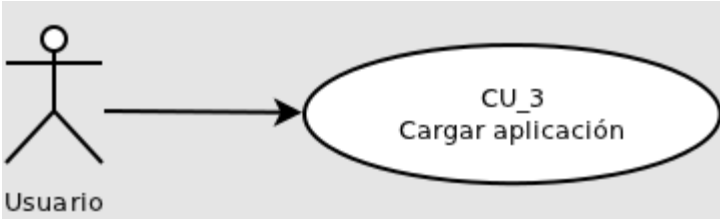

Figura 14: caso de uso (cargar aplicación)

Identificador	CU_4: añadir aplicación a favoritas.
Descripción	• El usuario elige una categoría en el menú de aplicaciones y dentro de la categoría hace click sobre la aplicación deseada. • Después de hacer click sobre el nombre de la aplicación, se mostrará una descripción de dicha aplicación. • Al hacer click en el enlace [Añadir a favoritas] añadirá la aplicación a su lista de favoritas.
Precondiciones	• El usuario ha iniciado sesión previamente en el sistema. • La aplicación no está ya en su lista de aplicaciones favoritas.
Postcondiciones	• La aplicación se ha añadido a su lista de aplicaciones favoritas.

Tabla 26: caso de uso (añadir aplicaciones favoritas)

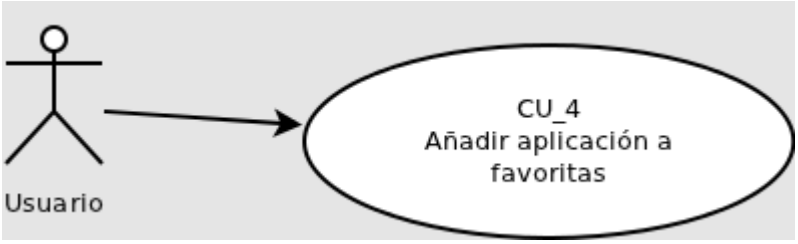

Figura 15: caso de uso (añadir aplicaciones favoritas)

Identificador	CU_5: quitar aplicación de favoritas.
Descripción	• El usuario elige una categoría en el menú de aplicaciones y dentro de la categoría hace click sobre la aplicación deseada. • Después de hacer click sobre el nombre de la aplicación, se mostrará una descripción de dicha aplicación. • Al hacer click encima del icono de la estrella verde la aplicación se quitará de su lista de favoritas.
Precondiciones	• El usuario ha iniciado sesión previamente en el sistema. • La aplicación está en su lista de aplicaciones favoritas.
Postcondiciones	• La aplicación se ha quitado de su lista de aplicaciones favoritas.

Tabla 27: caso de uso (eliminar aplicaciones favoritas)

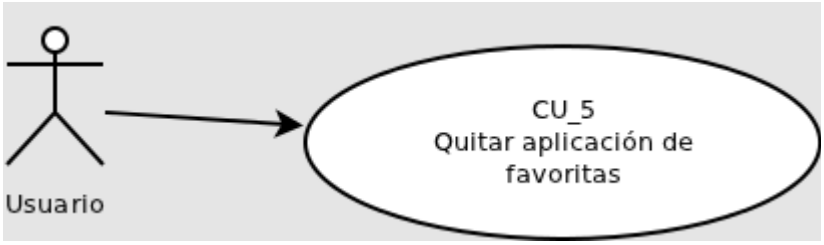

Figura 16: caso de uso (eliminar aplicaciones favoritas)

3.3 Diseño

Durante esta sección se discutirán los aspectos relacionados con la fase de diseño dentro de ciclo de desarrollo de software. Concretamente se van a comentar los detalles relacionados con el diseño del front-end y del back-end de Clouded Desktop.

3.3.1 Front-end

Primeramente haré una introducción de las herramientas que he utilizado para después centrarme en el diseño de la interfaz de usuario de Clouded Desktop (apartado 3.3.1.5).

3.3.1.1 HTML

HTML, siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>).

HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

3.3.1.1.1 Elementos

Los elementos son la estructura básica de HTML. Los elementos tienen dos propiedades básicas: atributos y contenido. Cada atributo y contenido tiene ciertas restricciones para que se considere válido al documento HTML. Un elemento generalmente tiene una etiqueta de inicio (p.ej. <nombre-de-elemento>) y una etiqueta de cierre (p.ej. </nombre-de-elemento>). Los atributos del elemento están contenidos en la etiqueta de inicio y el contenido está ubicado entre las dos etiquetas

(p.ej. <nombre-de-elemento atributo="valor">Contenido</nombre-de-elemento>). Algunos elementos, tales como <br>, no tienen contenido ni llevan una etiqueta de cierre. Debajo se listan varios tipos de elementos de marcado usados en HTML.

Estructura general de una línea de código en el lenguaje de etiquetas HTML.

El marcado estructural describe el propósito del texto. Por ejemplo, <h2>Golf</h2> establece «Golf» como un encabezamiento de segundo nivel, el cual se mostraría en un navegador de una manera similar al título «Marcado HTML» al principio de esta sección. El marcado estructural no define cómo se verá el elemento, pero la mayoría de los navegadores web han estandarizado el formato de los elementos. Un formato específico puede ser aplicado al texto por medio de hojas de estilo en cascada.

El marcado presentacional describe la apariencia del texto, sin importar su función. Por ejemplo, negrita indica que los navegadores web visuales deben mostrar el texto en negrita, pero no indica qué deben hacer los navegadores web que muestran el contenido de otra manera (por ejemplo, los que leen el texto en voz alta). En el caso de negrita e itálica, existen elementos que se ven de la misma manera pero tienen una naturaleza más semántica: énfasis fuerte y énfasis. Es fácil ver cómo un lector de pantalla debería interpretar estos dos elementos. Sin embargo, son equivalentes a sus correspondientes elementos presentacionales un lector de pantalla no debería decir más fuerte el nombre de un libro, aunque éste esté en itálicas en una pantalla. La mayoría del marcado presentacional ha sido desechada con HTML 4.0, en favor de hojas de estilo en cascada.

El marcado hipertextual se utiliza para enlazar partes del documento con otros documentos o con otras partes del mismo documento. Para crear un enlace es necesario utilizar la etiqueta de ancla <a> junto con el atributo href, que establecerá la dirección URL a la que apunta el enlace. Por ejemplo, un enlace a Clouded Desktop sería de la forma Clouded Desktop. También se pueden crear enlaces sobre otros objetos, tales como imágenes .

3.3.1.1.2 Atributos

La mayoría de los atributos de un elemento son pares nombre-valor, separados por un signo de igual «=» y escritos en la etiqueta de comienzo de un elemento, después del nombre de éste. El valor puede estar rodeado por comillas dobles o simples, aunque ciertos tipos de valores pueden

estar sin comillas en HTML (pero no en XHTML). De todas maneras, dejar los valores sin comillas es considerado poco seguro. En contraste con los pares nombre-elemento, hay algunos atributos que afectan al elemento simplemente por su presencia (tal como el atributo ismap para el elemento img).

3.3.1.2 CSS

El nombre hojas de estilo en cascada viene del inglés Cascading Style Sheets, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

Por ejemplo, el elemento de HTML <h1> indica que un bloque de texto es un encabezamiento y que es más importante que un bloque etiquetado como <H2>. Versiones más antiguas de HTML permitían atributos extra dentro de la etiqueta abierta para darle formato (como el color o el tamaño de fuente). No obstante, cada etiqueta <H1> debía disponer de la información si se deseaba un diseño consistente para una página y, además, una persona que leía esa página con un navegador perdía totalmente el control sobre la visualización del texto.

Cuando se utiliza CSS, la etiqueta <H1> no debería proporcionar información sobre cómo será visualizado, solamente marca la estructura del documento. La información de estilo, separada en una hoja de estilo, especifica cómo se ha de mostrar <H1>: color, fuente, alineación del texto, tamaño y otras características no visuales, como definir el volumen de un sintetizador de voz, por ejemplo.

La información de estilo puede ser adjuntada como un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style".

3.3.1.2.1 Ventajas

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.

- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local, que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o, incluso, a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil o ser "leída" por un sintetizador de voz.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño (siempre y cuando no se utilice estilo en línea).

3.3.1.3 Javascript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

3.3.1.3.1 jQuery

Figura 17: logo de jQuery

jQuery es una biblioteca o framework de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos.1 jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Las empresas Microsoft y Nokia anunciaron que incluirán la biblioteca en sus plataformas.2 Microsoft la añadirá en su IDE Visual Studio3 y la usará junto con los frameworks ASP.NET AJAX y ASP.NET MVC, mientras que Nokia los integrará con su plataforma Web Run-Time.

3.3.1.4 Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las

bibliotecas de clases de Java fueron desarrollados por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es).

Figura 18: logo de Java

3.3.1.4.1 Applets

Un applet Java es un applet escrito en el lenguaje de programación Java. Los applets de Java pueden ejecutarse en un navegador web utilizando la Java Virtual Machine (JVM), o en el AppletViewer de Sun.

Entre sus características podemos mencionar un esquema de seguridad que permite que los applets que se ejecutan en el equipo no tengan acceso a partes sensibles (por ej. no pueden escribir archivos), a menos que uno mismo le dé los permisos necesarios en el sistema; la desventaja de este enfoque es que la entrega de permisos es engorrosa para el usuario común, lo cual juega en contra de uno de los objetivos de los Java applets: proporcionar una forma fácil de ejecutar aplicaciones desde el navegador web.

En Java, un applet es un programa que puede incrustarse en un documento HTML, es decir en una página web. Cuando un navegador carga una página web que contiene un applet, este se descarga en el navegador web y comienza a ejecutarse. Esto permite crear programas que cualquier usuario puede ejecutar con tan solo cargar la página web en su navegador.

El navegador que carga y ejecuta el applet se conoce en términos genéricos como el "contenedor" de los applets. El kit de desarrollo de software para Java 2 (J2SDK) 1.4.1 incluye un

contenedor de applets, llamado appletviewer, para probar los applets antes de incrustarlos en una página web.

3.3.1.5 La interfaz de usuario de Clouded Desktop

A partir de la siguiente página se muestran unas capturas de las distintas pantallas que conforman la interfaz de usuario de Clouded Desktop y un mapa de navegación entre ellas (de forma apaisada para conservar la máxima resolución posible).

En la figura 19 se muestra la pantalla de entrada a Clouded Desktop. En caso de introducir algún dato incorrecto se denegará el acceso y se mostrará el correspondiente mensaje de error (figura 20). En la figura 21 puede verse la pantalla principal de Clouded Desktop con los iconos de las aplicaciones favoritas del usuario. (Estos iconos se pueden arrastrar para ordenarlos a nuestro gusto).

Al pinchar en un icono o en algún elemento del menú, se abre una ventana con una breve descripción de la aplicación (figura 22) En esta pantalla de descripción, arriba a la derecha hay un enlace que nos permite añadir la aplicación a nuestra lista de aplicaciones favoritas (figura 25) o eliminarla de ella (figura 24).

Si queremos acceder a nuestra cuenta de Dropbox bastará con pulsar sobre el enlace que se muestra en el pie de página e instantáneamente se abrirá una ventana flotante con la web de Dropbox para que iniciemos sesión (figura 23). Al pie de esta ventana se mostrará el nombre de usuario de la cuenta de Dropbox asociada a nuestra cuenta de Clouded Desktop.

En la ventana de descripción si pulsamos en el enlace CARGAR APLICACIÓN se abrirá una ventana flotante con el applet Java de NoMachine (figura 26). Bastará con pulsar el botón CONTINUE para que comience la carga de la aplicación (figura 27). Una vez cargada la aplicación, esta ventana se cerrará automáticamente y si la aplicación cargada está en nuestra lista de aplicaciones favoritas se resaltará en rojo el icono de la aplicación en ejecución. (Para no saturar el servidor de Clouded Desktop sólo se permite al usuario ejecutar una instancia de cada aplicación).

Por último, si intentamos cerrar sesión teniendo alguna aplicación de Clouded Desktop en ejecución, se nos alertará de ello y se nos dará la opción de cerrarlas todas pulsando un botón (figura 28 y 29). En dicho mensaje se recordará también al usuario que cierre su sesión de Dropbox si la hubiere abierto.

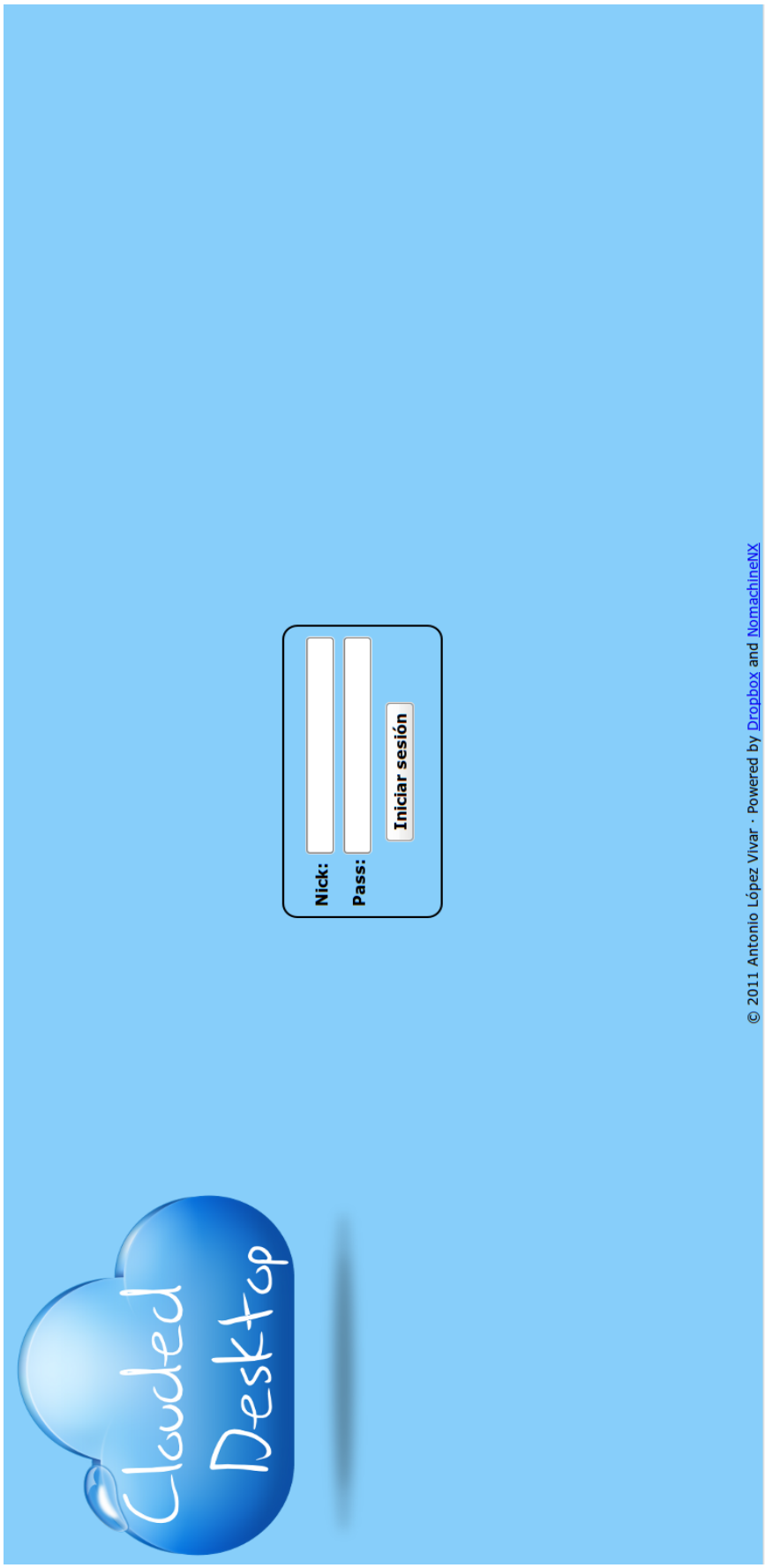

Figura 19: pantalla-CD1 (pantalla de login)

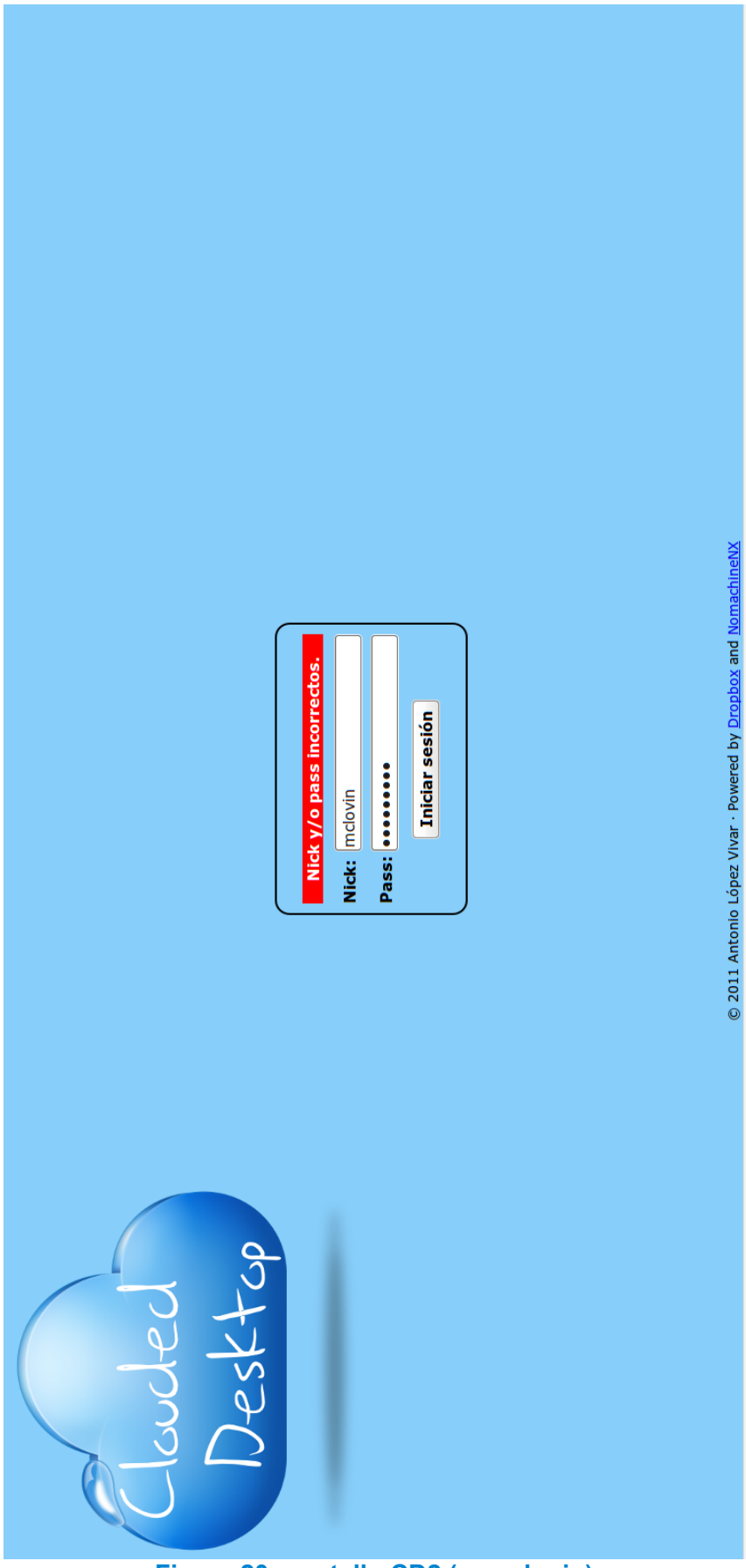

Figura 20: pantalla-CD2 (error login)

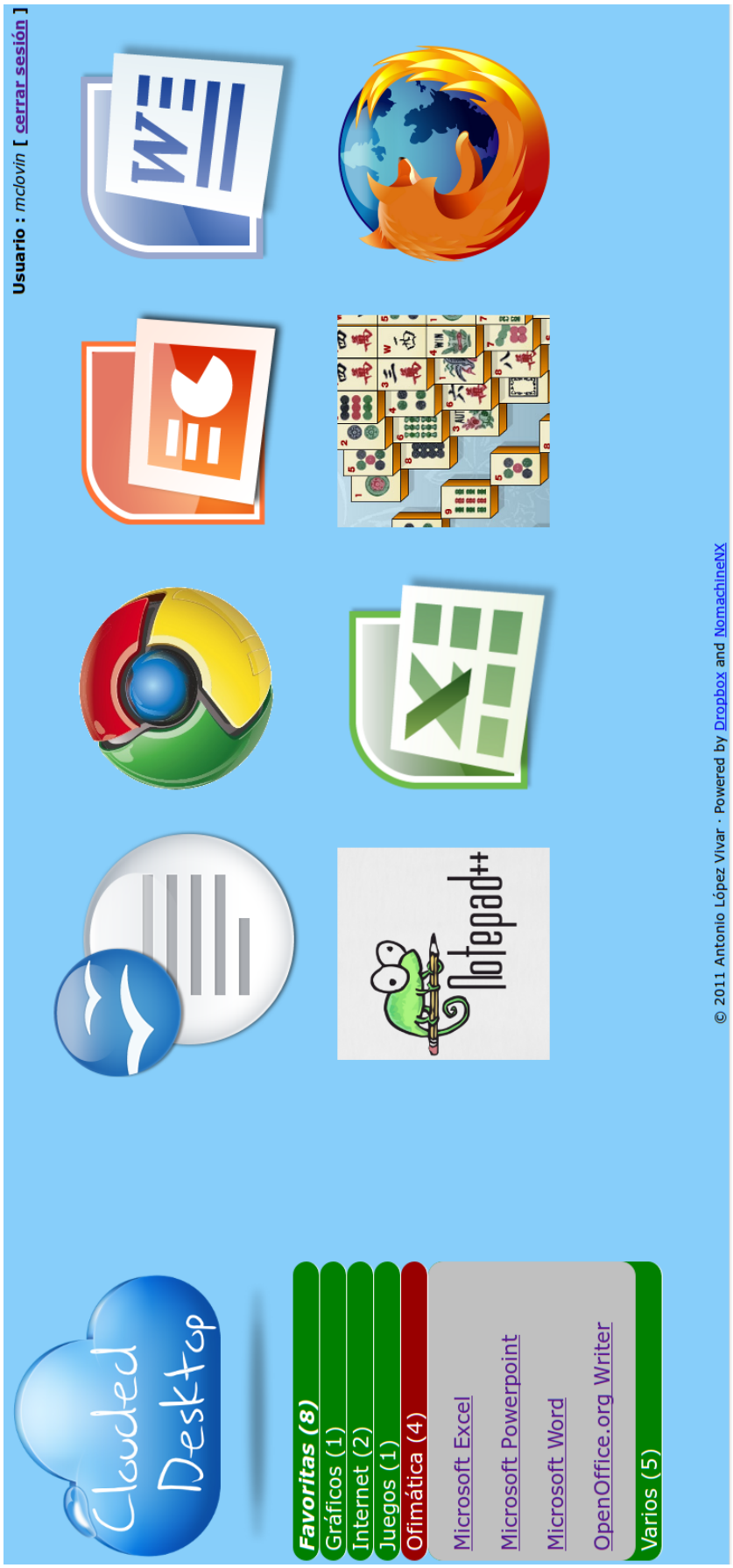

Figura 21: pantalla-CD3 (escritorio)

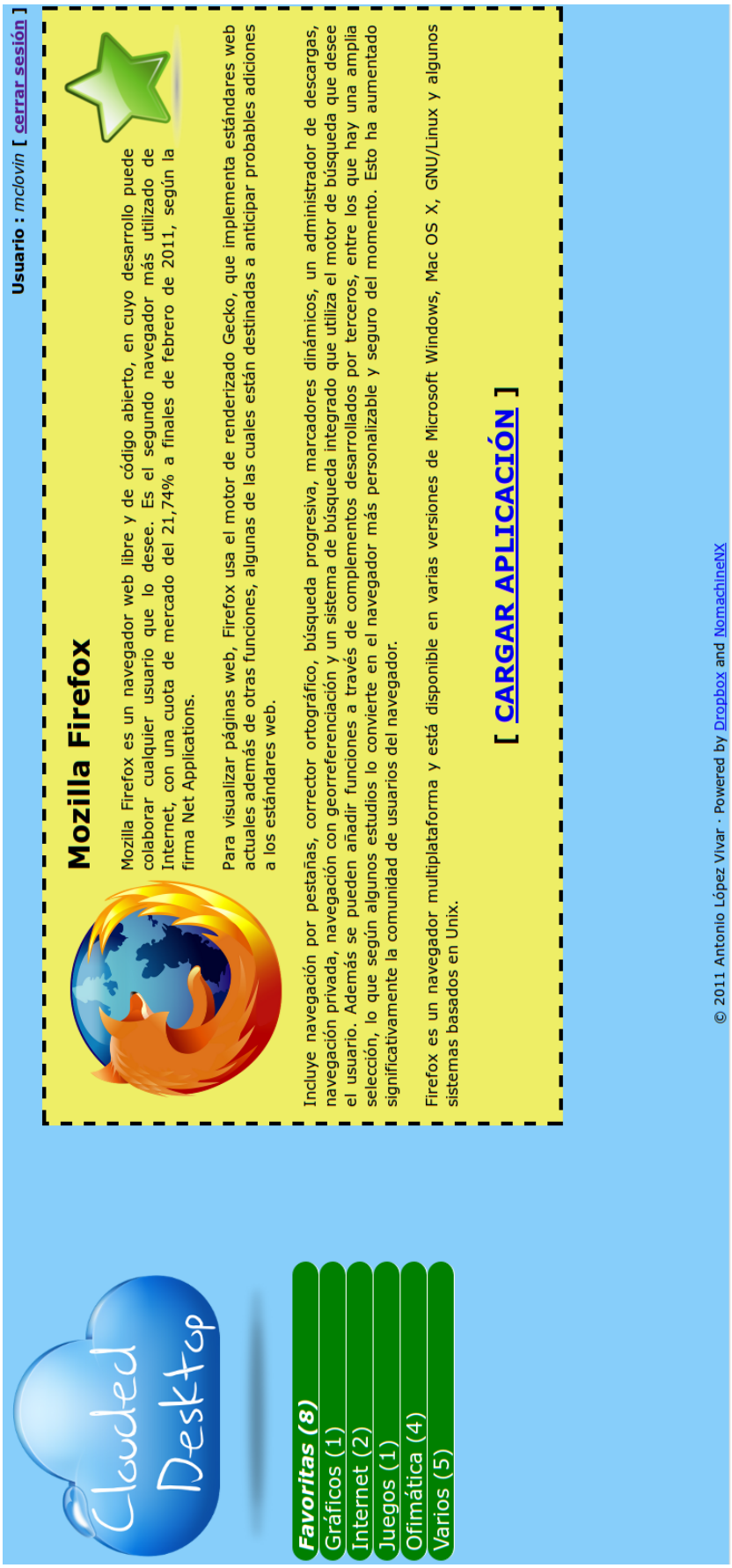

Figura 22: pantalla-CD4 (ventana informativa)

Figura 23: pantalla-CD5 (ventana de Dropbox)

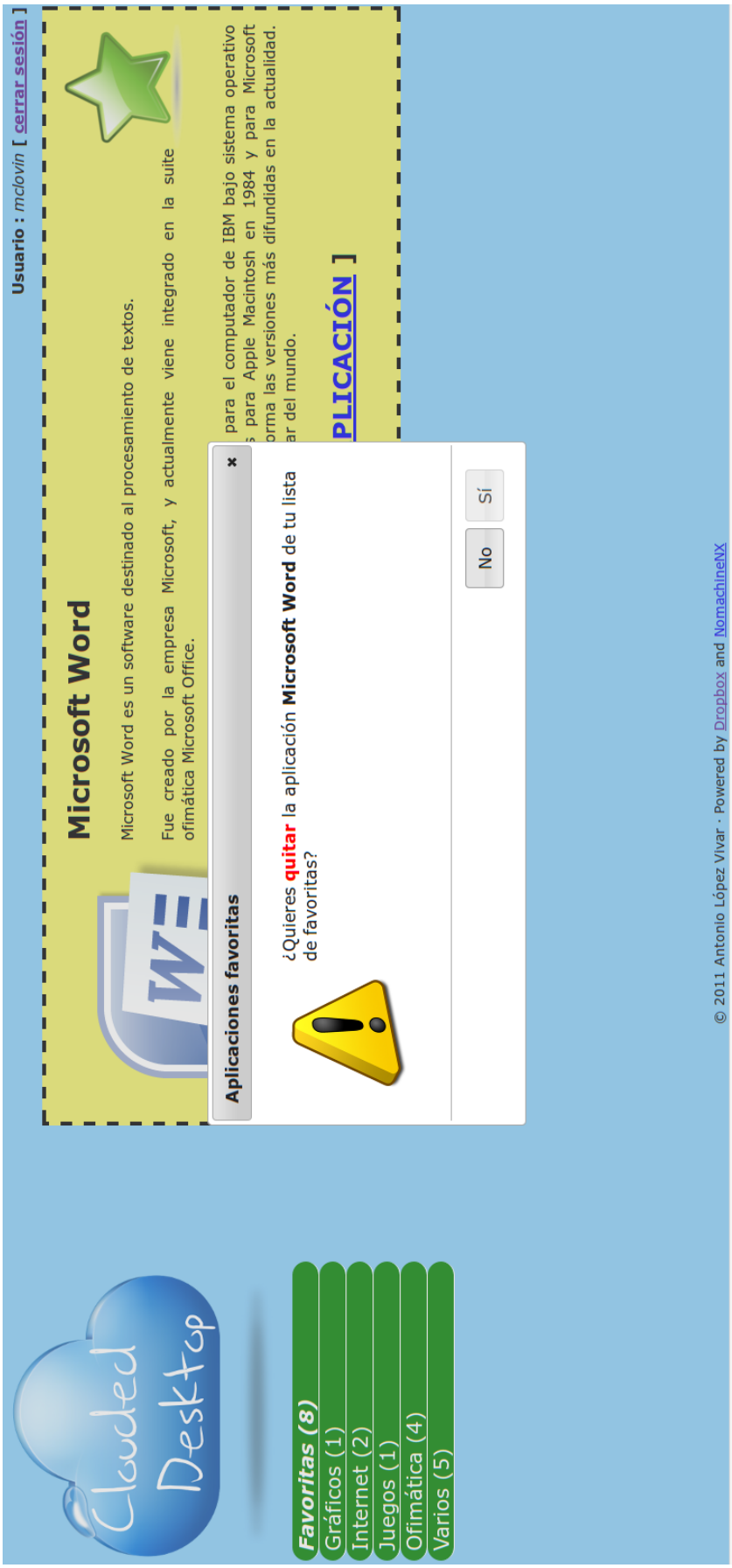

Figura 24: pantalla-CD6 (aviso eliminación favoritas)

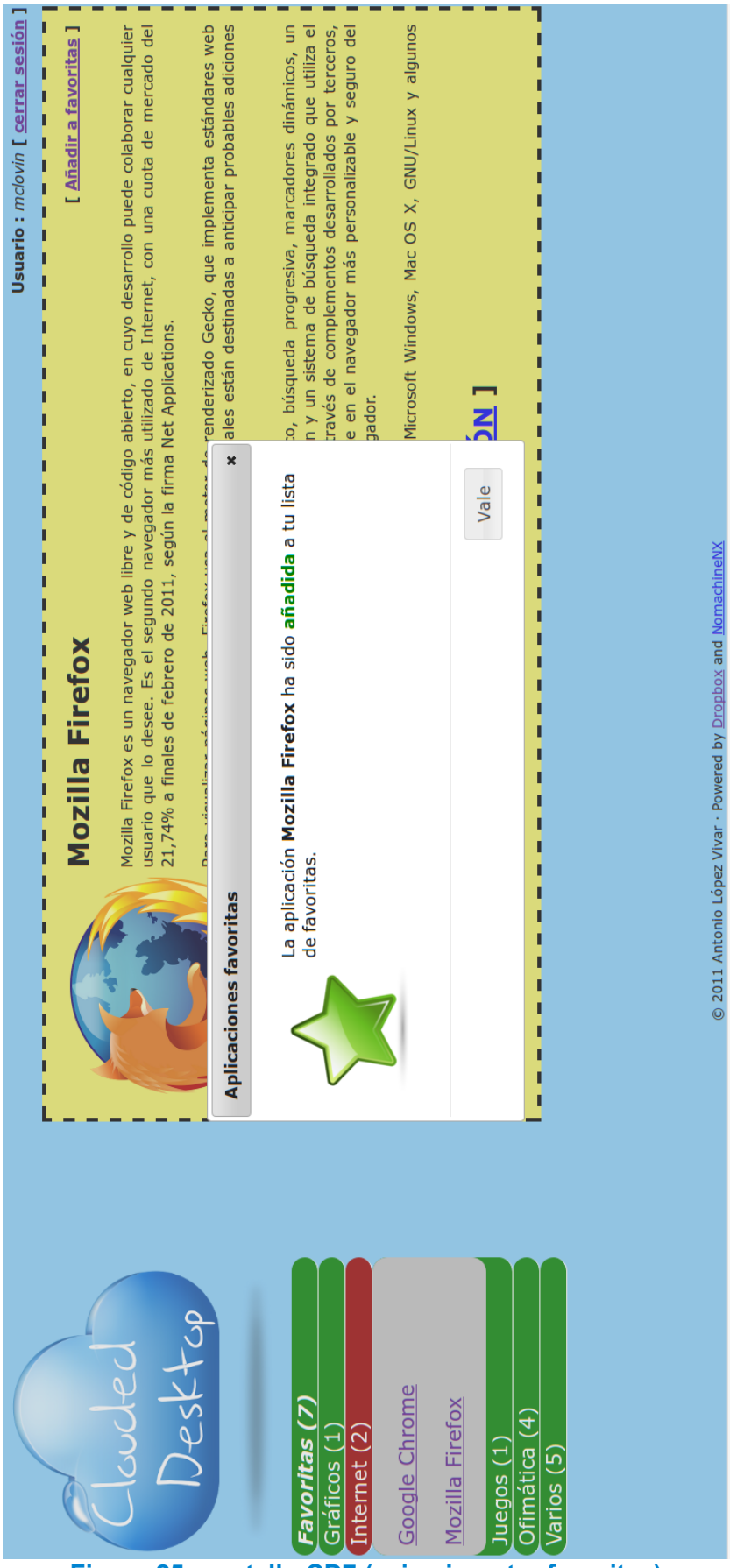

Figura 25: pantalla-CD7 (aviso insertar favoritas)

Figura 26: pantalla-CD8 (applet NoMachine preparado)

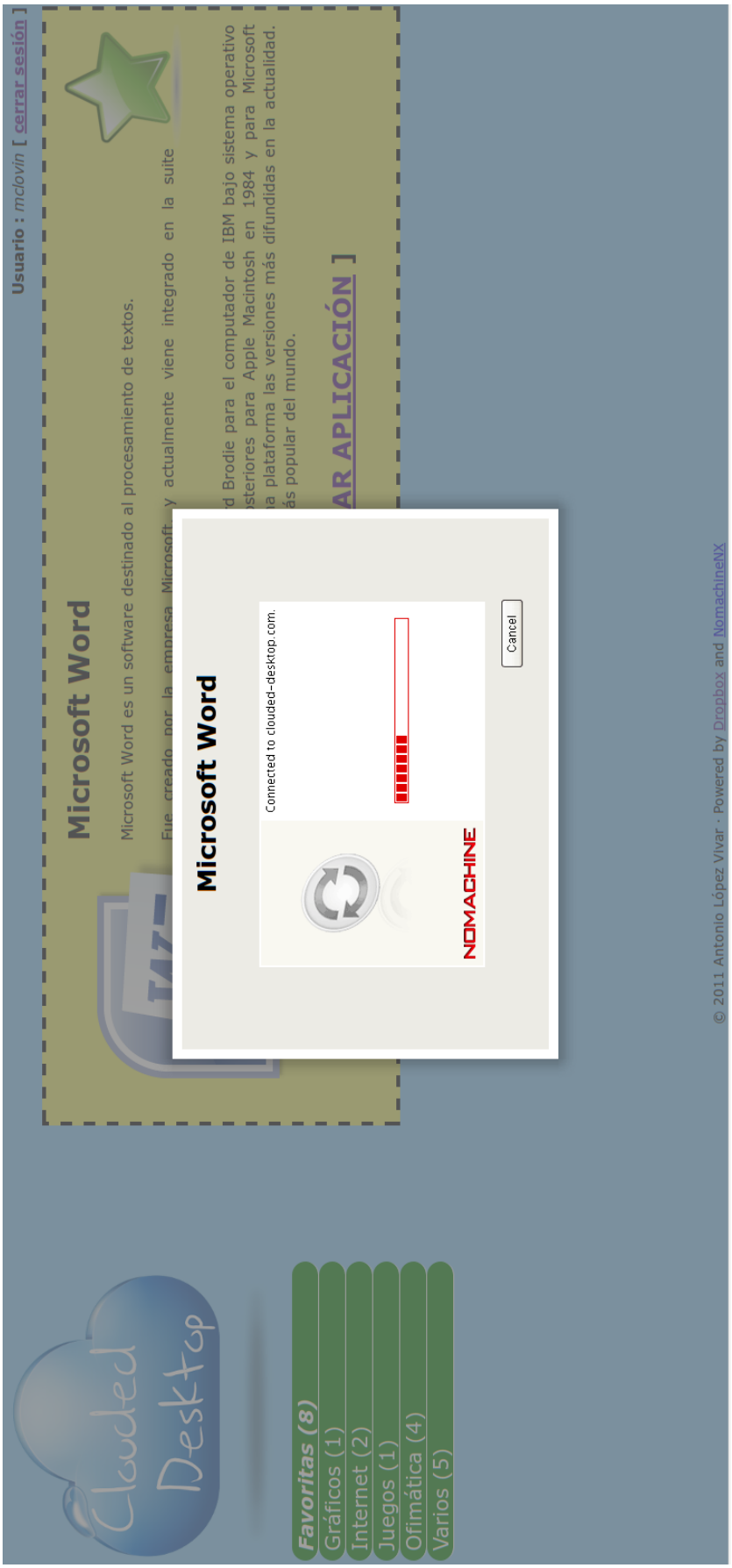

Figura 27: pantalla-CD9 (applet NoMachine cargando)

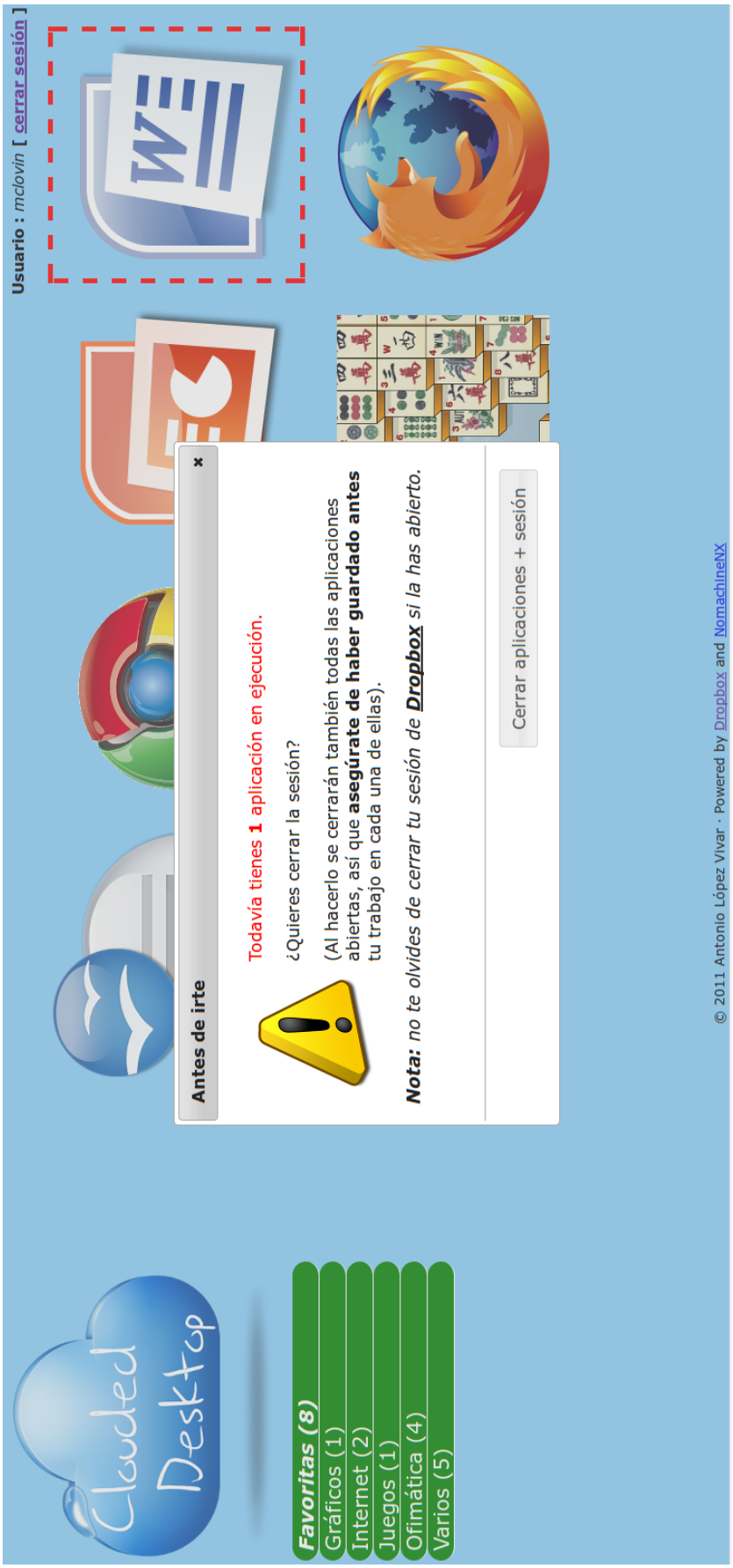

Figura 28: pantalla-CD10 (aviso aplicaciones ejecución)

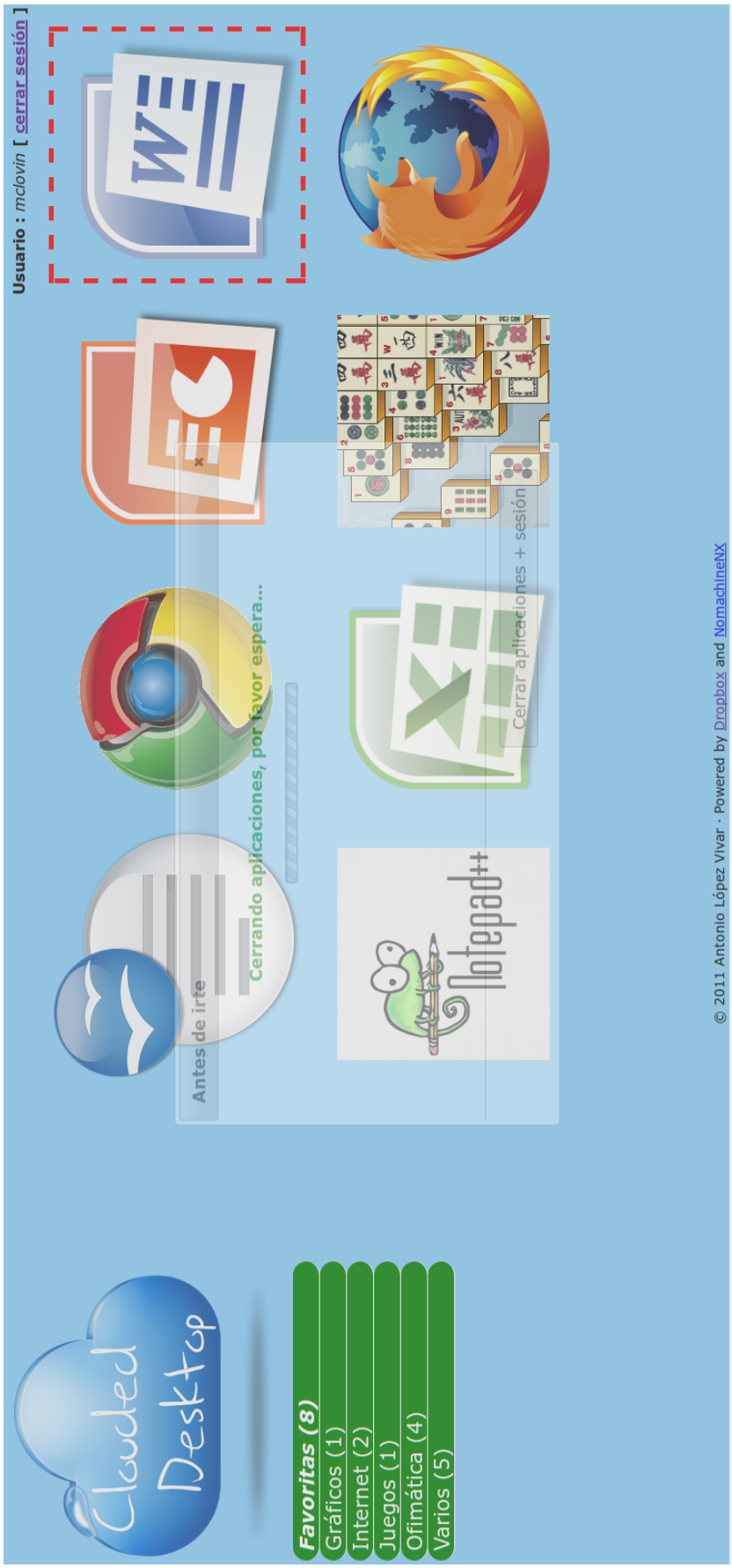

Figura 29: pantalla-CD11 (aviso cierre de aplicaciones)

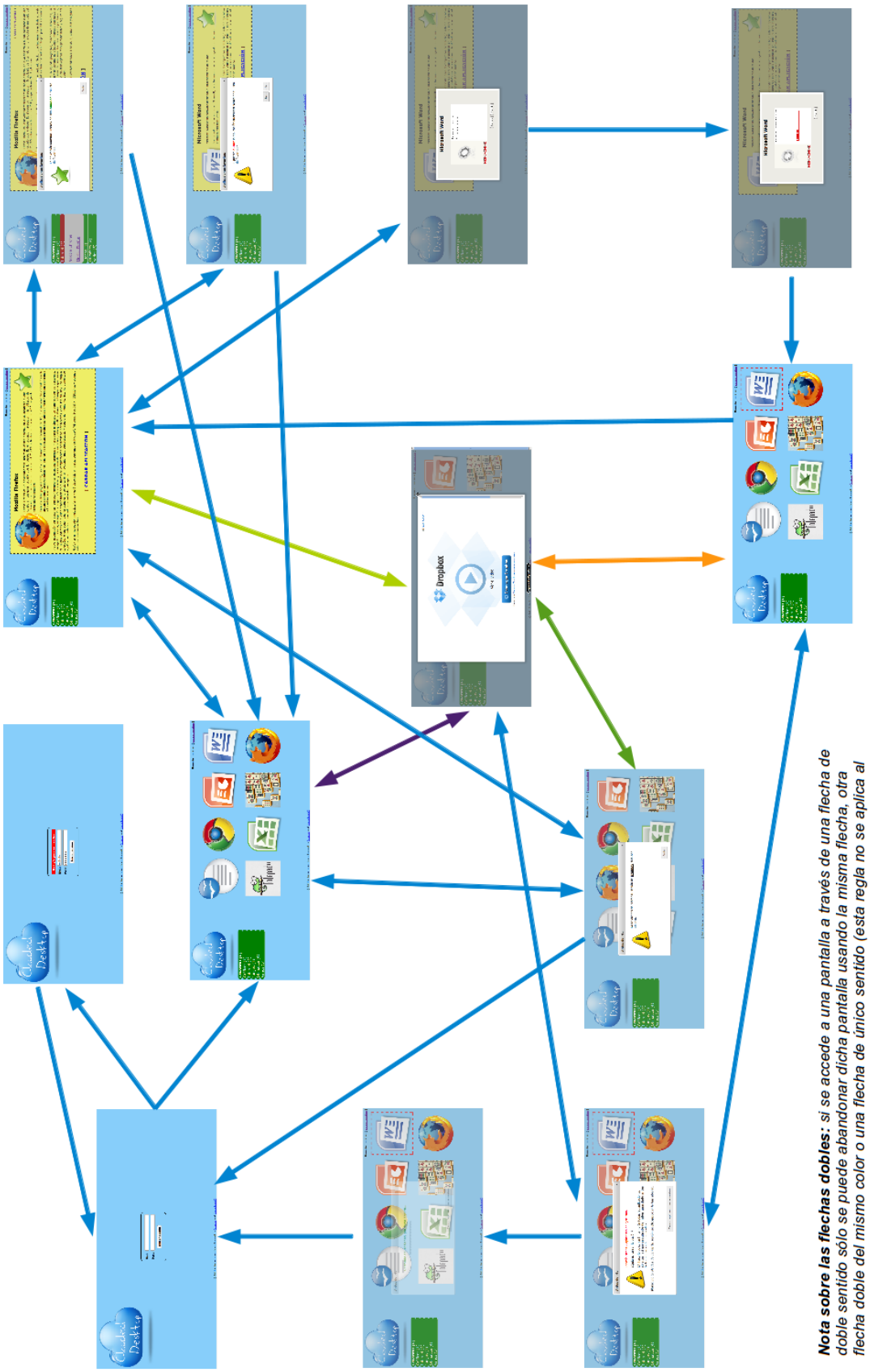

Figura 30: esquema de navegación por pantallas de Clouded Desktop

3.3.2 Back-end

Al igual que en el front-end primeramente haré una introducción del software y los lenguajes de programación que he utilizado para luego centrarme en el diseño del back-end (apartado 3.3.2.2.3) y en el de la base de datos (apartado 3.3.2.3.3.1 en adelante). Finalmente hablaré un poco de Dropbox y de por qué lo elegí para el proyecto.

3.3.2.1 Servidor web: Apache.

Un servidor web es un programa que está diseñado para transferir hipertextos, páginas web o páginas HTML (HyperText Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. El programa implementa el protocolo HTTP (HyperText Transfer Protocol) que pertenece a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa.

El Servidor web se ejecuta en un ordenador manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error. A modo de ejemplo, al teclear clouded-desktop.com en nuestro navegador, éste realiza una petición HTTP al servidor de dicha dirección. El servidor responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo exhibe en pantalla. Como vemos con este ejemplo, el cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Además de la transferencia de código HTML, los Servidores web pueden entregar aplicaciones web. Éstas son porciones de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Hay que distinguir entre:

- **Aplicaciones en el lado del cliente:** el cliente web es el encargado de ejecutarlas en la máquina del usuario. Son las aplicaciones tipo Java "applets" o Javascript: el servidor proporciona el código de las aplicaciones al cliente y éste, mediante el navegador, las ejecuta. Es necesario, por tanto, que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones (también llamadas scripts). Comúnmente, los navegadores permiten

ejecutar aplicaciones escritas en lenguaje JavaScript y java, aunque pueden añadirse más lenguajes mediante el uso de plugins.

- **Aplicaciones en el lado del servidor:** el servidor web ejecuta la aplicación; ésta, una vez ejecutada, genera cierto código HTML; el servidor toma este código recién creado y lo envía al cliente por medio del protocolo HTTP. Las aplicaciones de servidor muchas veces suelen ser la mejor opción para realizar aplicaciones web. La razón es que, al ejecutarse ésta en el servidor y no en la máquina del cliente, éste no necesita ninguna capacidad añadida, como sí ocurre en el caso de querer ejecutar aplicaciones JavaScript o java. Así pues, cualquier cliente dotado de un navegador web básico puede utilizar este tipo de aplicaciones.

El hecho de que HTTP y HTML estén íntimamente ligados no debe dar lugar a confundir ambos términos. HTML es un lenguaje de marcas y HTTP es un protocolo.

3.3.2.1.1 Apache HTTP Server

El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de Internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, a patchy server (un servidor "parcheado").

Figura 31: logo de Apache

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que

ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Entre sus características principales se pueden destacar:

- Modular
- Código abierto
- multiplataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)

3.3.2.1.1.1 Módulos

La arquitectura del servidor Apache es muy modular. El servidor consta de una sección core y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web. Algunos de estos módulos son:

- *mod_ssl* - Comunicaciones Seguras vía TLS.
- *mod_rewrite* - reescritura de direcciones (generalmente utilizado para transformar páginas dinámicas como php en páginas estáticas html para así engañar a los navegantes o a los motores de búsqueda en cuanto a cómo fueron desarrolladas estas páginas).
- *mod_dav* - Soporte del protocolo WebDAV (RFC 2518).
- *mod_deflate* - Compresión transparente con el algoritmo deflate del contenido enviado al cliente.

- *mod_auth_ldap* - Permite autentificar usuarios contra un servidor LDAP.
- *mod_proxy_ajp* - Conector para enlazar con el servidor Jakarta Tomcat de páginas dinámicas en Java (servlets y JSP).

El servidor de base puede ser extendido con la inclusión de módulos externos entre los cuales se encuentran:

- *mod_cband* - Control de tráfico y limitador de ancho de banda.
- *mod_perl* - Páginas dinámicas en Perl.
- *mod_php* - Páginas dinámicas en PHP.
- *mod_python* - Páginas dinámicas en Python.
- *mod_rexx* - Páginas dinámicas en REXX y Object REXX.
- *mod_ruby* - Páginas dinámicas en Ruby.
- *mod_aspdotnet* - Páginas dinámicas en .NET de Microsoft (Módulo retirado).
- *mod_mono* - Páginas dinámicas en Mono
- *mod_security* - Filtrado a nivel de aplicación, para seguridad.

3.3.2.1.1.2 Uso

Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

Según una investigación realizada por *Elbanis H.R* Apache es el componente de servidor web en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP/Perl/Python (y ahora también Ruby).

Este servidor web es redistribuido como parte de varios paquetes propietarios de software, incluyendo la base de datos Oracle y el IBM WebSphere application server. Mac OS X integra Apache como parte de su propio servidor web y como soporte de su servidor de aplicaciones WebObjects. Es soportado de alguna manera por Borland en las herramientas de desarrollo Kylix y

Delphi. Apache es incluido con Novell NetWare 6.5, donde es el servidor web por defecto, y en muchas distribuciones GNU/Linux.

Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde una computadora personal hacia Internet. Un usuario que tiene Apache instalado en su escritorio puede colocar arbitrariamente archivos en la raíz de documentos de Apache, desde donde pueden ser compartidos.

Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de previsualizar y probar código mientras éste es desarrollado.

3.3.2.1.1.3 Licencia

La licencia de software bajo la cual el software de la fundación Apache es distribuido es una parte distintiva de la historia de Apache HTTP Server y de la comunidad de código abierto. La Licencia Apache permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

La Free Software Foundation no considera a la Licencia Apache como compatible con la versión 2 de la GNU General Public License (GPL), en la cual el software licenciado bajo la Apache License no puede ser integrado con software distribuido bajo la GPL:

Este es software libre pero es incompatible con la GPL. La Apache Software License es incompatible con la GPL porque tiene un requerimiento específico que no está incluido en la GPL: tiene ciertos casos de terminación de patentes que la GPL no requiere. No consideramos que dichos casos de terminación de patentes son inherentemente una mala idea, pero a pesar de ello son incompatibles con la GNU GPL.

Sin embargo, la versión 3 de la GPL incluye una provisión (Sección 7e) que le permite ser compatible con licencias que tienen cláusulas de represalia de patentes, incluyendo a la Licencia Apache.

El nombre Apache es una marca registrada y puede ser sólo utilizada con el permiso expreso del dueño de la marca.

3.3.2.2 Programación: PHP y Bash scripting.

Aunque este no es un proyecto de programación "puro", ha sido necesario programar bastantes scripts en PHP y BASH que sirvieran de "pegamento" entre la interfaz de usuario de Clouded Desktop y el servidor NX.

3.3.2.2.1 PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

Figura 32: logo de PHP

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores, el número de sitios en PHP ha compartido algo de su preponderante sitio con otros nuevos lenguajes no tan poderosos desde agosto de 2005. Este mismo sitio web de Wikipedia está desarrollado en PHP. Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web. La versión más reciente de PHP es la 5.3.5, del 6 de enero de 2011.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de página web, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando la extensión PHP-Qt o PHP-GTK. También puede ser usado desde la línea de órdenes, de la misma manera como Perl o Python pueden hacerlo; a esta versión de PHP se la llama PHP-CLI (Command Line Interface).

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor Web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. El programa está liberado bajo la licencia GNU y actúa como un servidor Web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP esta disponible para Microsoft Windows, GNU/Linux, Solaris, y MacOS X.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como GNU/Linux o Mac OS X) y Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

PHP es una alternativa a las tecnologías de Microsoft ASP y ASP.NET (que utiliza C# VB.NET como lenguajes), a ColdFusion de la compañía Adobe (antes Macromedia), a JSP/Java de Oracle, y a CGI/Perl. Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, existe además un IDE (entorno de desarrollo integrado) comercial llamado Zend Studio. Recientemente, CodeGear (la división de lenguajes de programación de Borland) ha sacado al mercado un entorno integrado de desarrollo para PHP, denominado Delphi for PHP. También existen al menos un par de módulos para Eclipse, uno de los IDE más populares.

3.3.2.2.1.1 Ventajas

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun estando dirigido a alguna en particular, el programador puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (o MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.

3.3.2.2.1.2 Inconvenientes

Como es un lenguaje que se interpreta en ejecución para ciertos usos puede resultar un inconveniente que el código fuente no pueda ser ocultado. La ofuscación es una técnica que puede dificultar la lectura del código pero no la impide y aparte en ciertos casos representa un costo en tiempos de ejecución.

3.3.2.2.2 Bash

Bash es un programa informático cuya función consiste en interpretar órdenes. Está basado en la shell de Unix y es compatible con POSIX. Fue escrito para el proyecto GNU y es el intérprete de comandos por defecto en la mayoría de las distribuciones de GNU/Linux. Su nombre es un acrónimo de Bourne-Again Shell (otro shell bourne) — haciendo un juego de palabras (born-again significa renacimiento) sobre el Bourne shell (sh), que fue uno de los primeros intérpretes importantes de Unix.

Hacia 1978 Bourne era el intérprete distribuido con la versión del sistema operativo Unix Versión 7. Stephen Bourne, por entonces investigador de los Laboratorios Bell, escribió la versión original de Bourne. Brian Fox escribió bash en 1987. En 1990, Chet Ramey se convirtió en su principal desarrollador. Bash es el intérprete predeterminado en la mayoría de sistemas GNU/Linux, además de Mac OS X Tiger, y puede ejecutarse en la mayoría de los sistemas operativos tipo Unix. También se ha llevado a Microsoft Windows por el proyecto Cygwin.

3.3.2.2.2.1 Sintaxis

La sintaxis de órdenes de bash es un superconjunto de instrucciones basadas en la sintaxis del intérprete Bourne. La especificación definitiva de la sintaxis de órdenes de bash, puede encontrarse en el bash Reference Manual distribuido por el proyecto GNU.

La mayoría de los shell scripts (guiones de órdenes) Bourne pueden ejecutarse por bash sin ningún cambio, con la excepción de aquellos scripts de shell Bourne que hacen referencia a variables especiales de Bourne o que utilizan una orden interna de Bourne. La sintaxis de órdenes de bash incluye ideas tomadas desde el Korn Shell (ksh) y el C Shell (csh), como la edición de la línea de órdenes, el historial de órdenes, la pila de directorios, las variables $RANDOM y $PPID, y la sintaxis de substitución de órdenes POSIX: $(...). Cuando se utiliza como un intérprete de

órdenes interactivo, bash proporciona autocompletado de nombres de programas, nombres de archivos, nombres de variables, etc, cuando el usuario pulsa la tecla TAB

3.3.2.2.3 Scripts programados para Clouded Desktop

A continuación se listan todos los scripts programados para Clouded Desktop, ordenados alfabéticamente junto a una descripción de su función (sin entrar en detalles de implementación).

Nombre del script	Función
app.php	Crea al vuelo el fichero temporal .nxs que usará el applet Java de NoMachine y genera el código HTML del popup que contendrá el applet.
app_redirect.php	Redirige a la página principal de Clouded Desktop cuando el applet de NoMachine es cerrado.
index.php	Genera todo el código HTML de la página principal de Clouded Desktop.
login.php	Autentica al usuario.
logout.php	Desautentica al usuario.
ajax_php/app_desc.php	Genera el código HTML que contiene la descripción de una aplicación dada.
ajax_php/cargadas.php	Devuelve un entero con el total de aplicaciones cargadas por el usuario.
ajax_php/check_app_load.php	Comprueba si una aplicación dada ha sido cargada por el usuario.
ajax_php/clean_temp.php	Elimina todos los ficheros de sesión temporales del usuario.
ajax_php/close_all_app.php	Cierra todas las aplicaciones del usuario llamando a *nxforce_close*. Después llama a *kill_all.sh* primero con la señal SIG_TERM y luego con SIG_KILL para garantizar que no queden procesos del usuario en ejecución en el servidor.
ajax_php/fav.php	Añade/borra una aplicación de la lista de favoritas del usuario.
ajax_php/kill_all.sh	Manda una señal a todas las aplicaciones del usuario.
***ajax_php/nxforce_close**	Pequeñísima aplicación programada en C encargada de ordenar al servidor NX el cierre de todas las aplicaciones pertenecientes a un usuario determinado (tiene el bit setuid activado para poder ejecutarse como root por usuarios "normales").
js/varios.php	Genera al vuelo un fichero con funciones JavaScript comunes.
loaders/cambiapass.sh	Cambia la contraseña UNIX del usuario.
loaders/pre_tank.sh	Lanza el demonio de Dropbox, elimina el fichero temporal .nxs del applet NoMachine, ejecuta *tank.php* y una vez éste finaliza (porque la aplicación lanzada por el usuario ha terminado), se queda esperando hasta que todos los ficheros creados o modificados hayan sido sincronizados con la cuenta de Dropbox.
loaders/tank.php	Es el cargador maestro. Se encarga de lanzar la aplicación elegida por el usuario teniendo en cuenta si es una aplicación de GNU/Linux o de Microsoft Windows (en cuyo caso la ejecutará a través de Wine). Cada vez que es ejecutado este script se cambia la contraseña UNIX del usuario por cuestiones de seguridad.
loaders/wait_dropbox.sh	Espera a que el "demonio" de Dropbox termine sus operaciones actuales.
require/mysql.php	Conecta con la base de datos de Clouded Desktop.
require/popurri.php	Contiene distintas funciones usadas por el resto de scripts.

Tabla 28: scripts programados para Clouded Desktop

3.3.2.3 Base de datos: MYSQL.

A continuación haré una pequeña introducción a los SGBD, centrándome en la opción escogida para mi proyecto y finalmente mostraré un análisis de la base de datos utilizada en Clouded Desktop.

3.3.2.3.1 SGBD

Los sistemas de gestión de bases de datos (en inglés database management system, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información:** Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.
- **Independencia:** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- **Consistencia:** En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa una realidad determinada que tiene determinadas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones.
- **Seguridad:** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Manejo de transacciones:** Una transacción es un programa que se ejecuta como una sola operación. Esto quiere decir que luego de una ejecución en la que se produce una falla es el

mismo que se obtendría si el programa no se hubiera ejecutado. Los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

- **Tiempo de respuesta:** Lógicamente, es deseable minimizar el tiempo que el SGBD demora en proporcionar la información solicitada y en almacenar los cambios realizados.

3.3.2.3.1.1 Ventajas

Proveen facilidades para la manipulación de grandes volúmenes de datos (ver objetivos). Entre éstas:

- Simplifican la programación de equipos de consistencia.
- Manejando las políticas de respaldo adecuadas, garantizan que los cambios de la base serán siempre consistentes sin importar si hay errores correctamente, etc.
- Organizan los datos con un impacto mínimo en el código de los programas.
- Disminuyen drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores.

Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos.

3.3.2.3.1.2 Inconvenientes

- Típicamente, es necesario disponer de una o más personas que administren de la base de datos, en la misma forma en que suele ser necesario en instalaciones de cierto porte disponer de una o más personas que administren los sistemas operativos. Esto puede llegar a incrementar los costos de operación en una empresa. Sin embargo hay que balancear este aspecto con la calidad y confiabilidad del sistema que se obtiene.
- Si se tienen muy pocos datos que son usados por un único usuario por vez y no hay que realizar consultas complejas sobre los datos, entonces es posible que sea mejor usar una planilla de cálculo.
- **Complejidad:** los software muy complejos y las personas que vayan a usarlo deben tener conocimiento de las funcionalidades del mismo para poder aprovecharlo al máximo.

- Tamaño: la complejidad y la gran cantidad de funciones que tienen hacen que sea un software de gran tamaño, que requiere de gran cantidad de memoria para poder correr.
- Coste del hardware adicional: los requisitos de hardware para correr un SGBD por lo general son relativamente altos, por lo que estos equipos pueden llegar a costar gran cantidad de dinero.

3.3.2.3.2 MySQL

Figura 33: logo de MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones escrito en C y C++. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

3.3.2.3.2.1 Historia

SQL (Lenguaje de Consulta Estructurado) fue comercializado por primera vez en 1981 por

IBM, el cual fue presentado a ANSI y desde entonces ha sido considerado como un estándar para las bases de datos relacionales. Desde 1986, el estándar SQL ha aparecido en diferentes versiones como por ejemplo: SQL:92, SQL:99, SQL:2003. MySQL es una idea originaria de la empresa opensource MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

Michael "Monty" Widenius en la década de los 90 trató de usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido y flexible para sus necesidades. Esto lo llevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

El nombre de MySQL procede de la combinación de My, hija del cofundador Michael "Monty" Widenius, con el acrónimo SQL (según la documentación de la última versión en inglés). Por otra parte, el directorio base y muchas de las bibliotecas usadas por los desarrolladores tenían el prefijo My.

El nombre del delfín de MySQL es Sakila y fue seleccionado por los fundadores de MySQL AB en el concurso “Name the Dolphin”. Este nombre fue enviado por Ambrose Twebaze, un desarrollador de software libre africano. Es el de una ciudad de Arusha, Tanzania, cerca de Uganda (lugar de origen de Ambrose).

3.3.2.3.2.2 Lenguajes de programación

Existen varias APIs que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (via dbExpress), Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, Ruby,Gambas, REALbasic (Mac y GNU/Linux), (x)Harbour (Eagle1), FreeBASIC, y Tcl; cada uno de estos utiliza una API específica. También existe una interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL. También se puede acceder desde el sistema SAP, lenguaje ABAP.

3.3.2.3.2.3 Aplicaciones

MySQL es muy utilizado en aplicaciones web, como Drupal o phpBB, en plataformas

(GNU/Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Sea cual sea el entorno en el que va a utilizar MySQL, es importante adelantar monitoreos sobre el desempeño para detectar y corregir errores tanto de SQL como de programación

3.3.2.3.3 La base de datos de Clouded Desktop

A continuación se muestra el modelado de la base de datos, donde están reflejadas de forma gráfica las tablas utilizadas y las relaciones existentes entre ellas.

3.3.2.3.3.1 Modelo E:R (simplificado)

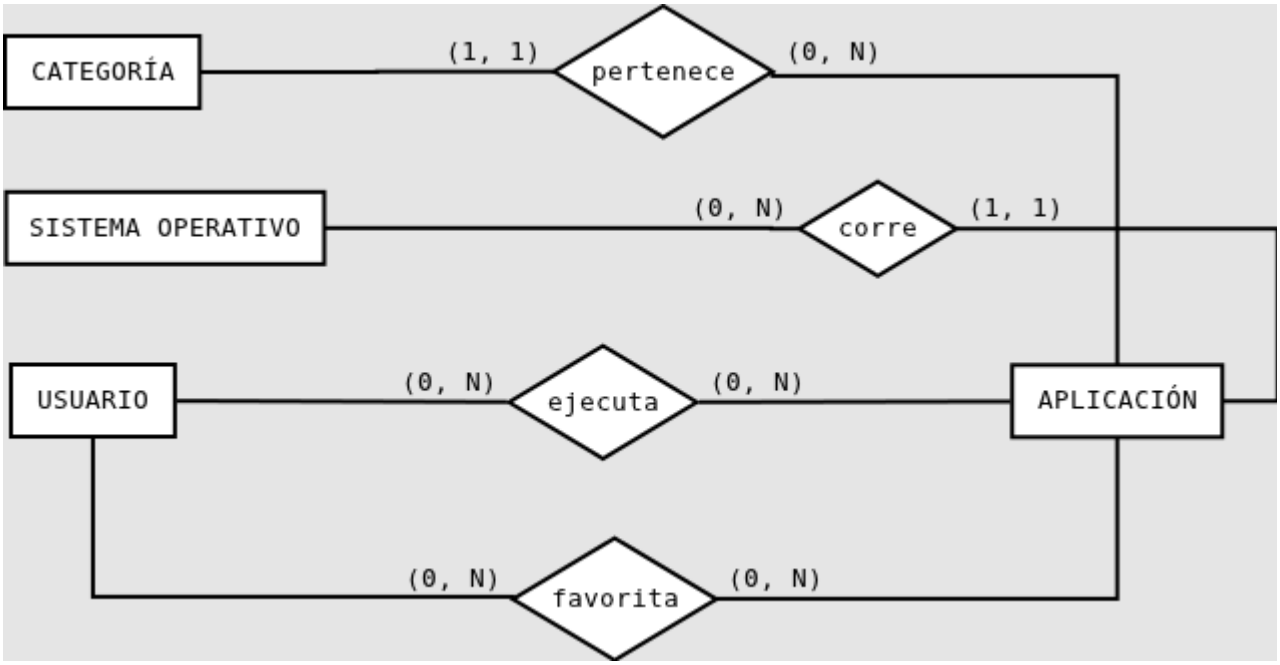

Figura 34: diagrama E:R de la base de datos de Clouded Desktop

3.3.2.3.3.2 Diagrama relacional

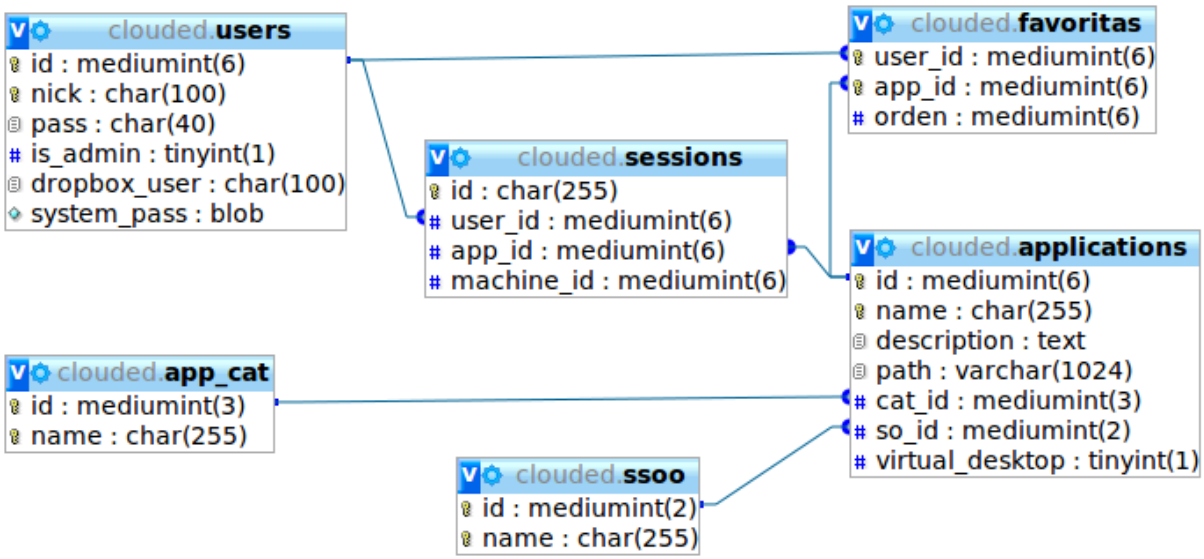

Figura 35: diagrama relacional de la base de datos de Clouded Desktop

3.3.2.3.3.3 Descripción de las tablas

users	listado de los usuarios de Clouded Desktop.
Id (PK)	identificador numérico de usuario, de uso interno al sistema.
nick (unique)	alias breve utilizado por el usuario en el sistema (case sensitive y sin espacios).
pass	contraseña del usuario para autenticarse en Clouded Desktop (hash SHA-1).
is_admin	campo booleano que indica si el usuario tiene privilegios de administración.
dropbox_user	email usado por el usuario para sincronizar los ficheros de Clouded Desktop con Dropbox (la contraseña no es necesario almacenarla ya que Dropbox genera un token de seguridad la primera vez que conecta que reusará en sesiones futuras).
system_pass	contraseña UNIX del usuario en el servidor/es donde corre Clouded Desktop (cifrada con AES).

Tabla 29: base de datos de Clouded Desktop (tabla users)

applications	listado de las aplicaciones disponibles en el sistema.
id (PK)	identificador numérico de la aplicación, de uso interno al sistema.
name (unique)	nombre de la aplicación.
description	descripción de la aplicación.
path	ruta absoluta para ejecutar la aplicación
cat_id (FK1)	categoría a la que pertenece la aplicación.
so_id (FK2)	sistema operativo al que pertenece la aplicación.
virtual_desktop	campo booleano usado internamente para mostrar la ventana de la aplicación.

Tabla 30: base de datos de Clouded Desktop (tabla aplicaciones)

sessions	la utilidad de esta tabla es llevar un registro de todas las aplicaciones que hay en ejecución en el sistema al mismo tiempo.
id (PK)	Identificador alfanumérico de la sesión, de uso interno al sistema.
user_id (FK1)	identificador numérico de usuario, de uso interno al sistema.
app_id (FK2)	identificador numérico de la aplicación, de uso interno al sistema.
machine_id	Identificador de la máquina donde se ejecutará la sesión (en el prototipo valdrá siempre 1, ya que sólo habrá una máquina).

Tabla 31: base de datos de Clouded Desktop (tabla sesiones)

favoritas	listado de las aplicaciones favoritas de todos los usuarios.
user_id (PK, FK1)	identificador numérico de la aplicación, de uso interno al sistema.
app_id (PK, FK2)	nombre de la aplicación.
orden	Indica la posición de la aplicación en el escritorio.

Tabla 32: base de datos de Clouded Desktop (tabla aplicaciones favoritas)

app_cat	listado de las categorías de aplicaciones
id (PK)	identificador numérico de la categoría, de uso interno al sistema.
name (unique)	nombre de la categoría.

Tabla 33: base de datos de Clouded Desktop (tabla categorías de aplicaciones)

ssoo	listado de los sistemas operativos en los que pueden “ejecutarse” las aplicaciones.
id (PK)	identificador numérico del sistema operativo, de uso interno al sistema.
name (unique)	nombre del sistema operativo.

Tabla 34: base de datos de Clouded Desktop (tabla sistemas operativos)

3.3.2.4 Aplicaciones para Microsoft Windows desde GNU/Linux: Wine

Wine (acrónimo recursivo en inglés para Wine Is Not an Emulator, que significa «Wine no es un emulador») es una reimplementación de la API de Win16 y Win32 para sistemas operativos basados en Unix. Permite la ejecución de programas diseñados para MS-DOS, y las versiones de Windows 3.11, 95, 98, ME, NT, 2000, XP, Vista y 7.

Wine provee de un conjunto de herramientas de desarrollo para portar código fuente de aplicaciones Windows a Unix además de un cargador de programas, el cual permite que muchas aplicaciones para Windows 2.0/3.x/9X/ME/NT/2000/XP/Vista y Win 7 se ejecuten sin modificarse en varios sistemas operativos unix-like como GNU/Linux, BSD, Solaris y Mac OS X

Figura 36: logo de Wine

El proyecto Wine comenzó en 1993, época de la versión 3.11 de Windows. El proyecto posiblemente se originó en discusiones en comp.os.linux. Los programadores Eric Youngdale y Bob Amstadt crearon su primera versión. La razón por la cual Wine no es un emulador es que los emuladores tienden a duplicar el entorno completo en el que un programa vive, incluyendo la simulación de una arquitectura de procesador determinada. Wine, por el contrario, implementa lo que podría ser llamado una capa de compatibilidad, la cual provee alternativas a las bibliotecas de Windows.

A mediados del año 2002, ya se contaba con una aplicación con más de 1 millón de líneas de código fuente escrito en lenguaje C y con un grupo de más de 300 programadores. El proyecto tuvo tiempos en los cuales no se avanzó lo suficiente, hasta que en el año 2003, aplicaciones muy extendidas en el entorno Windows como Microsoft Office e Internet Explorer fueron posibles de ser utilizadas en entornos Unix gracias a Wine.

El proyecto presenta grandes retos para los desarrolladores, al menos en parte debido a la incompleta documentación de la API de Windows. A pesar de que la mayoría de las funciones la API Win32 están correctamente documentadas, existen aún muchas áreas, como formatos de archivos y protocolos, para las cuales no existen especificaciones documentadas por parte de Microsoft.

Hacia comienzos de 2003, Wine podía ejecutar muchos programas populares, como Lotus Notes y algunas versiones de Microsoft Office, con comportamientos y estabilidad variables. El éxito del funcionamiento de cada aplicación depende del uso de bibliotecas dinámicas (DLL) de Windows.

La empresa de software Corel ayudó mucho al proyecto, empleando temporalmente a uno de los principales desarrolladores, Alexandre Julliard, junto con muchos otros programadores secundarios. Esta ayuda fue motivada por el porting de la suite ofimática de Corel a GNU/Linux.

Sin embargo, debido a dificultades económicas, el apoyo de Corel cesó o culminó.

Otras organizaciones han hecho esfuerzos comerciales para apoyar el proyecto, incluyendo CodeWeavers y Linspire. CodeWeavers ha desarrollado una versión comercial, cerrada y visualmente más atractiva, de Wine y la comercializa bajo el nombre CrossOver Office; además, colabora con el proyecto, ya sea financiándolo o con parches.

El desarrollo oficial de Wine está orientado hacia la correcta implementación de la API de Windows como un todo y aunque se encuentra un poco atrasado en estas áreas, desde su versión 1.0 de verano de 2008 es capaz de ejecutar con éxito y con pocos o escasos errores una gran variedad de aplicaciones diseñadas para Windows.

Otros proyectos que han incorporado código fuente de Wine son Rewind y ReactOS.

De acuerdo a un estudio realizado en Internet por DesktopLinux.com en 2006, la aplicación Wine es bastante usada con relación a las demás con una demanda de 33,5% de los encuestados, mientras que un 16,7% utiliza el VMware y apenas un 7% usa la aplicación CrossOver. En tanto la porción de los encuestados que no utiliza ninguno fue de un 39%.

El 17 de junio de 2008 el proyecto Wine lanzó la versión 1.0,2 la primera versión estable en quince años de desarrollo. Ésta presenta mejoras con relación a las versiones alfa y beta, de las cuales se puede mencionar un mejor soporte de ratón en los juegos, aplicaciones con uso de bibliotecas OpenGL, manejo de nuevos estados en Direct3D y mejora del sistema de audio, entre mejoras importantes...

3.3.2.5 Almacenamiento de ficheros en "la nube": Dropbox

Dropbox es un servicio de alojamiento de archivos multiplataforma en la nube, operado por la compañía Dropbox. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre computadoras y compartir archivos y carpetas con otros.1 Existen versiones gratuitas y de pago, cada una de las cuales con opciones variadas.

Figura 37: logo de Dropbox

3.3.2.5.1 Funcionalidad

El cliente de Dropbox permite a los usuarios dejar cualquier archivo en una carpeta designada. Ese archivo es sincronizado en la nube y en todas las demás computadores del cliente de Dropbox. Los archivos en la carpeta de Dropbox pueden entonces ser compartidos con otros usuarios de Dropbox o ser accedidos desde la página Web de Dropbox. Asimismo, los usuarios pueden grabar archivos manualmente por medio de un navegador web.

Si bien Dropbox funciona como un servicio de almacenamiento, se enfoca en sincronizar y compartir archivos. Tiene soporte para historial de revisiones, de forma que los archivos borrados de la carpeta de Dropbox pueden ser recuperados desde cualquiera de las computadoras sincronizadas. También existe la funcionalidad de conocer la historia de un archivo en el que se esté trabajando, permitiendo que una persona pueda editar y cargar los archivos sin peligro de que se puedan perder las versiones previas. El historial de los archivos está limitado a un período de 30 días, aunque existe una versión de pago que ofrece el historial ilimitado. El historial utiliza la tecnología de delta encoding. Para conservar ancho de banda y tiempo, si un archivo en una carpeta Dropbox de un usuario es cambiado, Dropbox solo carga las partes del archivo que son cambiadas cuando se sincroniza. Si bien el cliente de escritorio no tiene restricciones para el tamaño de los archivos, los archivos cargados por medio de la página Web están limitados a un máximo de 300 MB cada uno. Dropbox utiliza el sistema de almacenamiento S3 de Amazon para guardar los archivos y SoftLayer Technologies para su infraestructura de apoyo.

3.3.2.5.2 Cuentas

Dropbox permite elegir entre tres tipos de cuentas: la primera llamada "Basic" es gratuita; la segunda, llamada "Pro50", y la tercera, llamada "Pro100", son de pago. Las diferencias radican en la cantidad de espacio disponible para poder utilizar, mientras la "Basic" dispone de 2Gb, la "Pro50" dispone de 50Gb y la "Pro100" de 100Gb. Los precios de las dos cuentas de pago son 9,99$ al mes para la cuenta "Pro50" y 19,99$ al mes para la cuenta "Pro100".

3.3.2.5.3 Seguridad

La sincronización de Dropbox usa transferencias SSL y almacena los datos mediante el protocolo de cifrado AES-256.

3.3.2.5.4 Historia

Dropbox cuenta con más de 4 millones de usuarios y tiene presencia en 175 países. La compañía recibió capital semilla de Y-Combinator y de Sequoia Capital. Desde el día 18 de abril del 2011 Dropbox anuncia que está disponible en Español, además de Alemán, Japonés y Francés.

3.3.2.5.5 ¿Por qué Dropbox en Clouded Desktop?

Por cuestiones de tiempo decidí utilizar un servicio consolidado de sincronización y backup de ficheros ya existente en vez de implementarlo desde cero dentro de Clouded Desktop. El elegir Dropbox en vez de alguna otra alternativa fue debido a su popularidad y que por sus características la integración con Clouded Desktop es francamente buena. Al registrar un nuevo usuario de Clouded Desktop se tiene la opción de asociar una cuenta de Dropbox donde se subirá una copia de los ficheros creados/modificados desde cualquier aplicación de Clouded Desktop. Esto se hace forma completamente transparente para el usuario por un demonio que corre en el servidor y vigila el directorio del usuario en busca de cambios que enviar a Dropbox.

3.3.3 Ensamblando el "meccano"

Desde que un usuario conecta con Clouded Desktop hasta que comienza a utilizar la aplicación elegida se dan una serie de pasos que detallaré a continuación:

1. El usuario conecta mediante un navegador web con el servidor Apache de Clouded Desktop para solicitarle la página principal.
2. Una vez cargada la página de Clouded Desktop, el usuario inicia sesión introduciendo sus datos de acceso que son enviados mediante una petición HTTP POST al servidor Apache.
3. Si los datos de acceso son válidos, el usuario es autenticado y se le redirige a su página principal donde tiene todos los iconos de sus aplicaciones favoritas.
4. Una vez elegida una aplicación que cargar (mirando en el menú general o haciendo click sobre uno de los iconos de su "escritorio"), se abrirá una ventana con información sobre la aplicación en cuestión.
5. Si el usuario decide cargar la aplicación, pulsará en el botón que así lo indica que abrirá una ventana flotante con el applet Java de NoMachine "incrustado". Antes de que Apache le

envíe al navegador el fichero HTML que contiene dicho applet, en el servidor se crea al vuelo un fichero temporal XML con extensión .nxs que más tarde utilizará el applet para conectar con el servidor NX de Clouded Desktop. Este fichero contiene diversa información, como por ejemplo el nombre de usuario, la contraseña UNIX del usuario (ofuscada) y la ruta completa de la aplicación elegida entre otros.

6. Para terminar de cargar la aplicación bastará con pulsar el botón CONTINUE del applet y esperar a que se establezca la sesión con el servidor NX de Clouded Desktop.

7. Después de esto, el usuario podrá empezar a utilizar la aplicación o cargar otra distinta si lo desea.

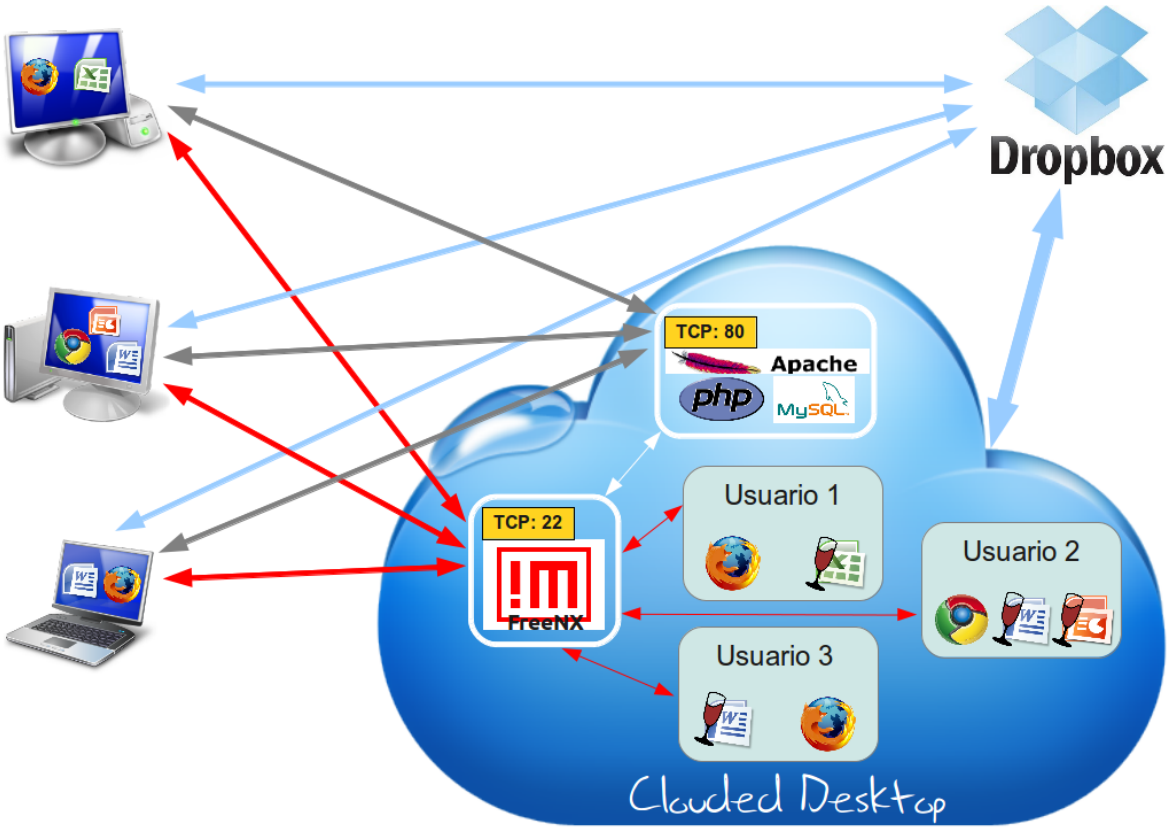

Figura 38: esquema de funcionamiento de Clouded Desktop

3.4 Implementación

En esta sección se van a comentar los aspectos de la implementación más interesantes y que como desarrollador han supuesto un mayor desafío o han resultado más llamativos o novedosos.

Para otras consideraciones relativas a la implementación se puede consultar el código fuente que se adjunta con este documento.

3.4.1 Generación del fichero XML temporal de sesión NX

```
function create_nx_session_file($user, $app)
{
    file_put_contents(($spath=TEMP_DIR.($sid=strtoupper(preg_replace('/[^a-z0-9]/i', '', $app['name'])).sha1(
    uniqid($user['nick']))).'.nxs'), strtr(file_get_contents(NX_SESSION_TEMPLATE_PATH), array('%clouded%app%'
     => implode(' ', array(NX_LOADER_PATH, $user['id'], $app['id'], $sid)), '%clouded%user%' => $user['nick'
    ], '%clouded%pass%' => trim(shell_exec(implode(' ', array(NX_PASS_SCRAMBLE_PATH, get_system_pass($user[
    'id'], $user['nick'].AES_KEY))))), '%clouded%host%' => HOST, '%clouded%port%' => NX_PORT,
    '%clouded%virtualdesktop%' => $app['virtual_desktop']?'true':'false')));

    chmod($spath, 0777);

    return $sid;
}
```

Figura 39: código para generar el fichero temporal .nxs

Cada vez que un usuario carga una aplicación desde la interfaz web de Clouded Desktop, se genera un fichero XML temporal de sesión .nxs que a continuación utilizará el applet Java de NoMachine para conectar al servidor NX de Clouded Desktop y comenzar la carga de la aplicación. Para generar este fichero temporal, se creó un fichero "plantilla" llamado *esqueleto.nxs* (ver el código fuente adjuntado al proyecto para más detalles) que contiene una serie de campos de la forma *%clouded%campo%* que serán sustituidos dentro de la función de la figura 39 por los valores actuales correspondientes en el momento de la llamada. Una vez cargada la aplicación, el fichero temporal será eliminado del servidor.

3.4.2 Carga de aplicaciones

El applet Java de NoMachine no carga directamente las aplicaciones, sino que se apoya en una serie de scripts escritos para Clouded Desktop que realizarán varias cosas además de cargar la aplicación.

3.4.2.1 Pre-cargador

El pre-cargador es el script al que llama el applet Java de NoMachine desde el navegador web pasándole tres argumentos, a saber: el id del usuario, el id de la aplicación y el id de la sesión NX. Como puede verse en la figura 40 el pre-cargador lanza el demonio de Dropbox, elimina el fichero temporal de sesión NX que ya no se necesita y llama al cargador principal. Una vez que el cargador devuelve la llamada (porque la aplicación ha sido cerrada), el pre-cargador llama a otro script (figura 41) que se quedará esperando hasta que el demonio de Dropbox termine de sincronizar cualquier fichero usado por el usuario.

```
1  #!/bin/bash
2
3  dropbox start -i
4  rm "/var/www/temp/$3.nxs"
5  php -f /var/www/loaders/tank.php $1 $2 $3
6  ./wait_dropbox.sh 2
7
```

Figura 40: código del pre-cargador de Clouded Desktop

```
1  #!/bin/bash
2
3  set -u
4
5  if [[ $(dropbox running) -ne 0 ]]
6  then
7  while [ "$(dropbox status)" != "Idle" ]; do
8      echo "Dropbox está ocupado. Me duermo $1 seg..."
9      sleep $1
10 done
11 fi
12
```

Figura 41: código de wait_dropbox.sh

3.4.2.2 Cargador

Antes de lanzar la aplicación, el cargador (figura 42) genera una contraseña UNIX nueva para el usuario, ya que la contraseña antigua se envió ofuscada en el fichero de sesión NX y un usuario malicioso podría extraerla de la caché del applet Java de NoMachine. Para cambiar la contraseña UNIX, se llama a otro script (figura 43). La particularidad del script para cambiar la contraseña UNIX es que esta tarea no puede automatizarse con un script "normal" ya que por motivos de seguridad el comando de Unix *passwd* no lee de la entrada estándar. Para solucionar

esto el script hace uso del comando *expect* que va leyendo la "respuesta" del comando *passwd* para enviarle la nueva contraseña.

```php
<?php

    if($argc==4 && is_numeric($argv[1]) && is_numeric($argv[2]))
    {
        require_once('/var/www/require/popurri.php');

        $user=get_user($argv[1]);
        $app=get_app($argv[2]);

        //Por seguridad, la contraseña UNIX de cada usuario cambia cada vez que se carga una aplicación nueva
        ya que un usuario malicioso podría obtenerla del fichero de sesión NX que enviamos antes.
        if(preg_match('/correctamente/i', exec('/var/www/loaders/cambiapass.sh '.$user['nick'].' '.
        get_system_pass($user['id'], $user['nick'].AES_KEY).' '.($new_spass=sha1(uniqid($user['nick'])))))
            set_system_pass($argv[1], $new_spass, $user['nick'].AES_KEY);

        add_session($argv[3], $argv[1], $argv[2]);

        switch($app['so_id'])
        {
            case 1:
                exec($app['path']);
                break;

            case 2:
                exec('env WINEPREFIX="/home/'.$user['nick'].'/.wine"');
                exec('wine '.str_replace('/root/', "/${user['nick']}/", $app['path']));
                break;
        }

        remove_session($argv[3]);
    }
    else
        echo "USO: ${argv[0]} <user_id> <app_id> <session_id>\n\n";
?>

```

Figura 42: código del cargador de Clouded Desktop

Una vez cambiada la contraseña, se crea una sesión nueva de Clouded Desktop y se procede a lanzar la aplicación. Si la aplicación elegida por el usuario es de GNU/Linux se lanza tal cual, pero en caso de haber elegido una aplicación de Windows se lanzará a través de *Wine* (configurando previamente una variable de entorno y actualizando el path como puede verse en la figura 42). Una vez se cierre la aplicación por el usuario, será eliminada de la base de datos la sesión y se devolverá el control al pre-cargador.

```
#!/usr/bin/expect

if {$argc != 3} {
    puts "Uso: $argv0 <usuario> <contraseña_antigua> <contraseña_nueva>"
    exit 1
}

set user [lindex $argv 0]
set old [lindex $argv 1]
set new [lindex $argv 2]

spawn su $user -c passwd

expect {

    "seña:" {
        send "$old\r"
        expect "UNIX:"
        send "$old\r"
        expect "UNIX:"
        send "$new\r"
        expect "UNIX:"
        send "$new\r"
    }

    "word:" {
        send "$old\r"
        expect "UNIX:"
        send "$old\r"
        expect "UNIX:"
        send "$new\r"
        expect "UNIX:"
        send "$new\r"
    }

    "UNIX:" {
        send "$old\r"
        expect "UNIX:"
        send "$new\r"
        expect "UNIX:"
        send "$new\r"
    }
}

expect eof
```

Figura 43: código del cambiador de contraseña

3.4.3 Cierre forzado de aplicaciones

Al cerrar sesión en Clouded Desktop se le da la opción al usuario de cerrar todas las aplicaciones que tenga en ejecución en ese momento dándole a un botón. Para implementar esta funcionalidad, se escribió el script de la figura 44 que en tres pasos cierra todas las aplicaciones de un usuario. Primero llama a *nxforce_close* (figura 45), que comunica al servidor NX que termine la sesión del usuario. La mayoría de las veces con esto bastaría para cerrar todos los procesos del usuario en el servidor, pero algunas veces el servidor NX no consigue cerrar todos los procesos del usuario. Por este motivo, se llama al script de la figura 46 que se encarga de enviar a todos los procesos que queden del usuario las señales SIG_TERM y SIG_KILL.

```php
<?php

    session_start();

    if(isset($_SESSION['user']))
    {
        require_once('../require/popurri.php');

        exec('./nxforce_close '.$_SESSION['user']['nick']);

        exec('./kill_all.sh -SIGTERM '.$_SESSION['user']['nick'].' '.($spass=get_system_pass($_SESSION['user']['id'],
        $_SESSION['user']['nick'].AES_KEY)));

        if(trim(exec('ps -u '.$_SESSION['user']['nick'].' -o "pid="')))
            exec('./kill_all.sh -SIGKILL '.$_SESSION['user']['nick'].' '.$spass);

        mysql_query('DELETE FROM sessions WHERE user_id='.$_SESSION['user']['id']);

        echo 1;
    }
    else
        echo -1;

?>
```

Figura 44: código de close_all_app.php

```c
#include <sys/types.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    setuid(0);

    if(argc==2)
        execlp("nxserver", "nxserver", "--force-terminate", argv[1], NULL);

    return 0;
}
```

Figura 45: código de nxforce_close.c

```
#!/usr/bin/expect

if {$argc != 3} {
    puts "Uso: $argv0 <signal> <usuario> <pass>"
    exit 1
}

set signal [lindex $argv 0]
set user [lindex $argv 1]
set pass [lindex $argv 2]

spawn su $user -c "kill $signal `ps -u $user -o \"pid=\"`"

expect {

    "seña:" { send "$pass\r" }

    "word:" { send "$pass\r" }
}

expect eof
```

Figura 46: código de kill_all.sh

Como curiosidad cabe mencionar que *nxforce_close* es un ejecutable con el bit SETUID activado para poder llamar a *nxserver* con permisos de root por el usuario de Apache. La razón de utilizar un programa en C en vez de un script de bash es que por diversos motivos de seguridad que no vienen al caso, no está permitido activar el bit SETUID en scripts.

4 Conclusiones y líneas futuras

4.1 Conclusiones

Desde el punto de vista de los objetivos, pienso que el proyecto ha sido un éxito. El prototipo desarrollado es ampliamente funcional y creo que demuestra perfectamente la idea buscada en un principio: separar al usuario la ejecución de aplicaciones informáticas de su instalación y mantenimiento además de permitirle hacerlo desde casi cualquier equipo con conexión a Internet.

Desde un punto de vista personal salgo también satisfecho de la experiencia, pues me ha servido para profundizar en el manejo de ciertos lenguajes y librerías que conocía de oídas pero no había tenido la ocasión de utilizar en ningún proyecto real, como por ejemplo Javascript y jQuery, además de valerme para refrescar conceptos adquiridos sobre varias materias de la carrera como bases de datos, diseño de software, programación, ingeniería de software y redes.

4.2 Líneas futuras

Como cualquier producto informático, Clouded Desktop es susceptible de ser mejorado y ampliado con nuevas funcionalidades. Quedarían por tanto varios aspectos pendientes de desarrollar como por ejemplo:

- Interfaz para administración: por cuestiones de tiempo, no se ha desarrollado la interfaz web para administrar Clouded Desktop. Sería interesante que un administrador pudiera controlar en tiempo real la carga del sistema, qué usuarios hay conectados, aplicaciones utilizadas, tiempo de uso, etc. así como dar de alta/baja usuarios, instalar/eliminar/modificar aplicaciones o restringir la ejecución de ciertas aplicaciones de forma remota.
- Escalabilidad: el prototipo de Clouded Desktop funciona en una sóla máquina, pero es evidente que esta solución no sería factible en un entorno real. En esa situación una posible solución sería tener una máquina para manejar el front-end, otra que haga de servidor de aplicaciones y después un número indeterminado de máquinas que se encarguen de ejecutar las aplicaciones con nxserver (comunicándose con el servidor de aplicaciones por NFS, por ejemplo). Clouded Desktop se encargaría de ir redirigiendo a los usuarios a distintas máquinas de forma transparente en función de la carga de CPU y memoria total del sistema.
- Sistema de sincronización de ficheros en "la nube" propio: a día de de hoy no existe un API HTTP de Dropbox que permita integrar en la interfaz de usuario de Clouded Desktop la gestión de los ficheros. Es por esto, que quizá sería buena idea implementar un sistema propio que se encargue de este asunto.
- Mejorar interfaz de usuario: sería bastante práctico añadir a la interfaz de usuario un buscador de aplicaciones que sugiera resultados a medida que se escribe.
- Mejorar soporte de aplicaciones Windows: aunque esto depende del proyecto Wine podrían estudiarse más a fondo otras vias de ejecución de aplicaciones Windows desde GNU/Linux.
- Añadir soporte para Mac: al menos para las aplicaciones exclusivas de esta plataforma.
- Aspectos legales: es evidente que en un entorno real de funcionamiento la cuestión de las licencias de las aplicaciones privativas instaladas en el servidor exigiría un estudio más profundo fuera del ámbito de este proyecto.

5 Planificación y presupuesto

5.1 Planificación

A continuación se va a detallar la planificación seguida para la elaboración del proyecto. Se ha de tener en cuenta que la dedicación a este proyecto no ha sido exclusiva, pues de forma simultánea se han desarrollado otras actividades que no han permitido dedicar todos los esfuerzos al mismo.

En la siguiente tabla se pueden ver las actividades realizadas durante la elaboración del proyecto indicándose la fecha de inicio y finalización de cada una de ellas así como la duración en días de cada actividad.

Actividad	Fecha inicio	Duración (días)	Fecha fin
Estudio de viabilidad	10/01/2011	7	17/1/2011
Documentación inicial	18/01/2011	15	2/2/2011
Análisis	3/2/2011	30	5/3/2011
Diseño	6/3/2011	42	17/4/2011
Implementación	17/4/2011	60	16/6/2011
Pruebas	17/5/2011	30	16/6/2011
Memoria del proyecto	12/5/2011	50	1/7/2011

Tabla 35: planificación de actividades del proyecto

5.2 Presupuesto

En esta sección se detallará el presupuesto estimado para el proyecto, detallando los gastos de personal, equipo y otros gastos.

Los gastos de personal se han calculado teniendo en cuenta los siguientes aspectos:

- Se han dedicado cuatro horas por día al proyecto.
- No se contabilizan los siguientes días festivos:

  19/03/2011, 22/04/2011, 1/5/2011, 2/5/2011, 15/5/2011 y 23/6/2011

Teniendo en cuenta estas consideraciones y la planificación del apartado anterior, los días de dedicación real al proyecto para cada actividad se muestran en la siguiente tabla:

Actividad	Duración real (días)
Estudio de viabilidad	7
Documentación inicial	15
Análisis	30
Diseño	41
Implementación	56
Pruebas	30
Memoria del proyecto	48

Tabla 36: planificación de actividades del proyecto (real)

Conociendo estos datos se pueden obtener los gastos de personal asociados al proyecto de la siguiente forma:

Coste de personal = ((Total días x Horas/día) / Dedicación hombre/mes) * Coste hombre/mes

Total días = 227 días

Horas/día = 4

Dedicación hombre mes = 160 horas

Coste hombre mes = 2523.38 €

Como resultado los gastos de personal alcanzan los CATORCE MIL TRESCIENTOS VEINTE CON DIECIOCHO CÉNTIMOS de euro.

Para la realización del proyecto se ha adquirido además el siguiente equipo, con un coste que queda detallado a continuación:

- Ordenador portátil Acer Aspire 5742G (341.53€ IVA NO incluido).

Teniendo en cuenta la siguiente fórmula para el cálculo de la amortización:

AMORTIZACIÓN = (A/B) * C * D

A: número de meses desde la fecha de facturación que el equipo es usado.

B: período de depreciación.

C: coste del equipo (sin IVA).

D: porcentaje de dedicación del equipo al proyecto.

el coste imputable al proyecto del equipo queda reflejado en la siguiente tabla:

Descripción	Coste (€)	% uso en el proyecto	Tiempo total de uso	Período de depreciación	Coste imputable
Portátil	341.53	100	7	60	39.85

Tabla 37: amortización equipo del proyecto

El coste total del proyecto queda reflejado en la siguiente tabla:

Concepto	Importe (€)
Personal	14320
Amortización	40
Costes indirectos (20%)	2872
Total (sin IVA)	17232
TOTAL (18% de IVA incluido)	**20334**

Tabla 38: costes del proyecto

Leganés, a 1 de julio de 2011

El ingeniero proyectista,

Fdo: Antonio López Vivar

Referencias y bibliografía

[1] García Abanades, Rubén. *Migración de un entorno web a Cloud Computing Amazon EC2* http://upcommons.upc.edu/pfc/bitstream/2099.1/8966/1/Memoria_PFC_Cloud_Computing.pdf

[2] http://www.codigoadicto.com/oraclesun-virtualbox/

[3] Gian Filippo Pinzari *NX X Protocol Compression*

http://www.nomachine.com/documents/pdf/NX-XProtocolCompression.pdf

[4] Protocolo NX http://en.wikipedia.org/wiki/NX_technology

[5] Servidor Web http://es.wikipedia.org/wiki/Servidor_web

[6] Apache http://es.wikipedia.org/wiki/Servidor_HTTP_Apache

[7] PHP http://es.wikipedia.org/wiki/PHP

[9] Spoon Streaming-White-Paper http://spoon.net/Server/Spoon-Streaming-White-Paper.pdf

[10] Spoon Server-User-Guide http://spoon.net/Server/Spoon-Server-User-Guide.pdf

[11] Spoon Studio-User-Guide http://spoon.net/Studio/Spoon-Studio-User-Guide.pdf

[12] Wine http://es.wikipedia.org/wiki/Wine

[13] HTML http://es.wikipedia.org/wiki/HTML

[14] Javascript http://es.wikipedia.org/wiki/Javascript

[15] jQuery http://es.wikipedia.org/wiki/Jquery

[16] MySQL http://es.wikipedia.org/wiki/Mysql

[17] Bash http://es.wikipedia.org/wiki/Bash

[18] Dropbox http://es.wikipedia.org/wiki/Dropbox

[19] Oviedo Expósito, Juan Manuel *Diseño e implementación de una aplicación Android para realizar presentaciones*